SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO – MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Krešimir Burić

LINEARNA TEMPORALNA LOGIKA

Diplomski rad

Voditelj rada:
prof. dr. sc.
Mladen Vuković

Zagreb, rujan 2014.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____ , predsjednik
2. _____ , član
3. _____ , član

Povjerenstvo je rad ocijenilo ocjenom _____ .

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Linearna temporalna logika (skraćeno LTL) je jedna vrsta modalne, odnosno temporalne logike. Osnovna karakteristika temporalne logike se očituje u specifičnosti njezinog jezika koji je pogodan za izražavanje koncepta vremena (lat. tempus – vrijeme, razdoblje, starost). Temporalna logika je jedna modalna logika čiji modalni operatori, tj. tzv. temporalni operatori omogućuju izražavanje vremenske komponente. Pojednostavljeno, moglo bi se reći da temporalna logika služi za proučavanje izjava koje sadrže komponentu vremena i čija istinitost se može mijenjati kroz vrijeme. Npr. promatrajući izjavu *„Jednog dana student K će diplomirati“* vidimo da npr. u klasičnoj logici sudova ne možemo adekvatno izraziti, a ni analizirati izrečenu izjavu jer promatrana izjava nije nužno univerzalno istinita ili lažna, nego njezina istinitost ovisi o vremenskoj komponenti.

U LTL–u se vrijeme modelira diskretno, tj. kao linearan niz stanja koji odgovaraju trenutcima u vremenu. Također, jedna od specifičnosti LTL-a je da se ne promatra prošlost tj. smatra se da je za određivanje budućnosti dovoljno poznavati samo sadašnjost. Točnije, smatra se da je prijelaz iz trenutnog stanja u buduće određen u potpunosti trenutnim stanjem i ne ovisi direktno o prošlim stanjima. Općenito, budućnost nije jedinstveno određena, poznavanjem trenutnog stanja nije nužno jednoznačno određeno sljedeće stanje. Stoga se obično promatra više nizova stanja koji predstavljaju različite budućnosti. Npr. izjava *„Sutra će padati kiša“* može biti istinita ali i ne mora biti istinita, tj. možemo zamisliti da postoji budućnost u kojoj će izjava biti istinita ali i da postoji budućnost u kojoj izjava neće biti istinita.

Začeci temporalne logike mogu se pronaći još u starogrčkoj filozofiji. No, oživljavanje temporalne logike kao formalne teorije počinje sredinom 20.-tog stoljeća radom logičara A. Priora[1] i S. Kripkea[2]. Pnueli [3] 1977.g. predlaže upotrebu temporalne logike za formalnu verifikaciju. Upotreba LTL-a u provjeri korektnosti rada računarskih sustava se pokazala kao jedna korisna metoda formalne verifikacije. U praksi, formalna verifikacija je od velike važnosti prilikom provjere korektnosti kritičnih sustava čije greške u radu mogu prouzrokovati velike materijalne, ekonomske štete ili čak ugroziti ljudske živote (npr. razni medicinski sustavi).

Sadržaj ovog rada je podijeljen u dva poglavalja. U prvom poglavlju se upoznajemo s jezikom linearne temporalne logike. Definiramo njezinu sintaksu i semantiku, te

[1] Arthur Prior - 1914. – 1969.
[2] Saul Kripke - 1940.
[3] Amir Pnueli - 1941. – 2009.

promatramo jedno proširenje linearne temporalne logike koje obuhvaća i koncept prošlosti. Uspoređivanjem izražajnosti ovih dviju logika, *opravdavamo* izostavljanje prošlosti iz LTL-a. Također, dajemo neke rezultate vezane za složenost pretvaranja formula proširenog LTL-a u ekvivalentne formule LTL-a, te odnose u veličini između tih formula.

U drugom poglavlju pokazujemo kako se LTL može koristiti prilikom verifikacije sustava. Definiramo pojam svojstva u kontekstu linearne temporalne logike. Ta svojstva predstavljaju formalizaciju i apstrakciju svojstava realnih sustava čiju korektnost želimo verificirati. U LTL-u svojstva izražavamo formulama. Dajemo jednu klasifikaciju svojstava i proučavamo karakterizacije različitih vrsta svojstava. Prepoznavanje vrste svojstva je bitno za verifikaciju sustava, budući da vrsta svojstva koje treba provjeriti značajno utječe na oblikovanje procesa verifikacije i odabir metode za verificiranje. Također, pokazujemo da LTL prihvaća dekompoziciju na tzv. svojstva sigurnosti i svojstva životnosti (eng. safety-liveness decomposition). Na kraju, konkretno kroz primjere pokazujemo kako se LTL primjenjuje prilikom verifikacije sustava, u sklopu jedne metode koja se zasniva na provjeri modela (tzv. *model checking*).

Poglavlje 1

Jezik linearne temporalne logike

Jezik linearne temporalne logike, odnosno kratko LTL-a, je sličan jeziku klasične logike sudova, s dodatkom temporalnih operatora koji omogućuju „gledanje“ u budućnost. U ovom poglavlju zadajemo jezik LTL-a. Prvo definiramo sintaksu, a zatim proučavamo semantiku, te mogućnosti koje nam daju temporalni operatori u izražajnosti LTL-a.

1.1 Sintaksa LTL-a

Osnovna struktura svakog jezika je njegov alfabet, odnosno skup simbola koje spajamo u riječi. Formalnije, **alfabet** je proizvoljan neprazan skup čije elemente nazivamo **simbol**i ili **znak**ovi. **Riječ** alfabeta je svaki konačan niz njegovih simbola.

Definicija 1.1.1. ***Alfabet LTL-a*** *je unija skupova* $A_1,\ A_2,\ A_3,\ A_4,\ A_5$ *pri čemu je:*

$A_1 = \{p_0,\ p_1,\ p_2\ ,\ \ldots\}$ *prebrojiv skup čije elemente nazivamo* ***propozicionalne varijable*** ;

$A_2 = \{\top, \bot\}$ *skup* ***logičkih konstanti*** *(istina i laž);*

$A_3 = \{\neg, \wedge, \vee, \rightarrow\}$ *skup* ***logičkih veznika*** ;

$A_4 = \{X,\ F,\ G,\ U,\ W,\ R\}$ *skup* ***temporalnih operatora***;

$A_5 = \{(\ ),\}$ *skup pomoćnih simbola (zagrade i zarez);*

Ponekad se u literaturi temporalni operatori nazivaju i temporalni veznici, no zapravo se radi o modalnim operatorima temporalne logike. Smatramo da su X, F, G unarni a U, W, R binarni operatori. Oznake[4] za temporalne operatore proizlaze iz načina njihove interpretacije, o čemu ćemo više reći malo kasnije. Sada definiramo najvažnije riječi alfabeta, tj. formule LTL-a.

Definicija 1.1.2. ***Atomarna formula*** *je svaka propozicionalna varijabla ili logička konstanta. Pojam* ***formule*** *definiramo rekurzivno*:

[4] Oznake za temporalne operatore imaju korijen u engleskim terminima koji upućuju na način njihove interpretacije: ne**X**t state, some **F**uture state, **G**lobally, **U**ntil, **W**eak-until, **R**elease.

a) *svaka atomarna formula je formula*
b) *ako su* ϕ *i* ψ *formule tada su i riječi* $(\neg\phi), (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi)$ *također formule*
c) *ako su* ϕ *i* ψ *formule tada su i riječi* $(X\ \phi), (F\ \phi), (G\ \phi), (\phi\ U\ \psi), (\phi\ R\ \psi), (\phi\ W\ \psi)$ *također formule*
d) *riječ alfabeta LTL-a je formula ako je nastala primjenom konačno mnogo koraka uvjeta a), b) i c)*

Napomena 1.1.3. *Formule LTL-a ćemo označavati grčkim slovima (npr.* ϕ, ψ, χ, ω *...). Za propozicionalne varijable ćemo upotrebljavati oznake* p, q, r, t *...*

Pisanje formula u sistemu zagrada može biti nepregledno. Uvođenjem prioriteta veznika omogućujemo izostavljanje pisanja zagrada kada one nisu potrebne, pri čemu ne generiramo dvosmislenosti. Izostavljanjem nepotrebnih zagrada formule postaju preglednije i lakše za pisati. Npr. $\Big(\big((F\ p) \wedge (G\ q)\big) \rightarrow (p\ W\ r)\Big)$ se može zapisati kao $F\ p \wedge G\ q \rightarrow p\ W\ r$. Najveći prioritet imaju unarni veznici, zatim dvomjesni temporalni, pa logički $\wedge$ i $\vee$, a najmanji prioritet ima kondicional $\rightarrow$. Pregledniji zapis prioriteta dan je sljedećom tablicom:

$\neg, X, F, G$
U, W, R
$\wedge, \vee$
$\rightarrow$

TABLICA 1.1: PRIORITETI LOGIČKIH VEZNIKA I TEMPORALNIH OPERATORA

Kao što simbole alfabeta spajamo u riječi tako možemo promatrati i spajanje riječi alfabeta. Ako su *a* i *b* riječi nekog alfabeta *A* tada je i riječ *ab* također riječ alfabeta *A*. Ta binarna operacija na skupu svih riječi A^* alfabeta *A* se naziva **konkatenacija**. Kažemo da je *b* **podriječ** riječi *a* ako postoje *c* i *d* tako da je riječ *a* nastala konkatenacijom riječi *c*, *b* i *d* tj. *a* je jednaka *cbd*. Kažemo da je formula ϕ **potformula** formule ψ ako je riječ ϕ podriječ od ψ.

Napomena 1.1.4. *U računarstvu se često za opisivanje sintakse jezika koristi Backus-Naurova forma (kratko: BNF) gdje je sintaksa definirana rekurzivno uz pomoć produkcijskih pravila. BNF definicija formula LTL-a je dana pravilom:*

$\phi ::= \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi\ U\ \phi) \mid (\phi\ R\ \phi) \mid (\phi\ W\ \phi) \mid (X\ \phi) \mid (F\ \phi) \mid (G\ \phi)$.

1.2 Modeli, putevi i istinitost

Jezik linearne temporalne logike koristimo za opisivanje dinamičkih sustava koji mijenjaju stanja. Propozicionalne varijable ćemo pridruživati stanjima, a formule LTL-a ćemo promatrati (tumačiti) isključivo u kontekstu tih sustava. Ti sustavi su određeni skupom stanja koje mogu poprimiti (statička komponenta) i skupom prijelaza između tih stanja (dinamička komponenta). Ovakve sustave nazivamo tranzicijskim sustavima. Sada dajemo formalnu definiciju pojma tranzicijski sustav.

Definicija 1.2.1. ***Tranzicijski sustav*** *je uređena trojka* $\mathcal{M} = (S, \mapsto, L)$ *pri čemu*

i) S *je skup čije elemente nazivamo stanja,*

ii) $\mapsto$ *je binarna relacija na* S *(tzv. tranzicijska relacija) za koju vrijedi da za svako stanje* $s \in S$ *postoji* $s' \in S$ *tako da* $s \mapsto s'$

iii) L *je proizvoljna funkcija s domenom* S *i kodomenom* $\mathcal{P}(A_1)$ *(gdje je* A_1 *skup propozicionalnih varijabli LTL-a), tj.* $L: S \longrightarrow \mathcal{P}(A_1)$

Tranzicijske sustave ćemo skraćeno nazivati **modelima**. Znači, model je jedna struktura koja se sastoji od skupa S nad kojim je definirana jedna binarna relacija i jedna funkcija. Elementi skupa S predstavljaju različita stanja sustava kojeg opisujemo. Binarna relacija $\mapsto$ opisuje dinamiku sustava definiranjem dozvoljenih prijelaza između stanja. Funkcija označavanja L *označava* pojedina stanja propozicionalnim varijablama LTL-a, tj. pridružuje im podskupove skupa A_1. Napomenimo da je tranzicijski sustav zapravo pojam koji se može definirati općenitije, neovisno o LTL-u, pri čemu funkcija označavanja tada stanjima pridružuje elemente nekog (proizvoljnog) skupa. Mi ćemo promatrati tranzicijske sustave isključivo u kontekstu LTL-a tj. tranzicijske sustave čija su stanja označena propozicionalnim varijablama LTL-a.

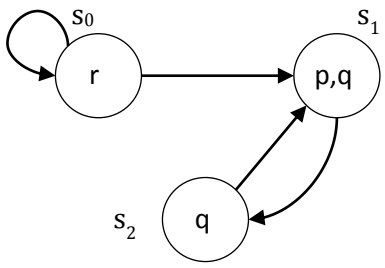
Na funkciju označavanja L možemo promatrati kao na analogon interpretacije iz logike sudova. Ona stanjima pridružuje propozicionalne varijable koje su *istinite* u tom stanju. Interpretacija u logici sudova je određivala koje su propozicionalne varijable *istinite* a koje *lažne.* No, sada imamo više stanja i u različitim stanjima propozicionalne varijable se mogu interpretirati na različite načine. Na L možemo zapravo promatrati kao na uniju interpretacija propozicionalnih varijabli po svim stanjima. S druge strane, restrikcija funkcije označavanja na pojedino stanje daje interpretaciju propozicionalnih varijabli u tom stanju. Pojam istinite formule u LTL-u ćemo precizno definirati kasnije. No, općenito LTL formulama želimo opisati *ponašanje* sustava dok on mijenja stanja. Stoga takve formule zapravo nema smisla interpretirati izolirano u pojedinim stanjima, nego ih interpretiramo na nizovima stanja.

Definicija 1.2.2. ***Put*** *u modelu* $\mathcal{M} = (S, \mapsto, L)$ *je beskonačni niz stanja* $s_0, s_1, s_2, \ldots$ *skupa S, takav da za svaki* $i \geq 0$ *vrijedi* $s_i \mapsto s_{i+1}$. *Put označavamo sa* $s_0 \mapsto s_1 \mapsto s_2 \mapsto \ldots$

Ako je π put u nekom modelu $\mathcal{M}$ onda pišemo $\pi \in \mathcal{M}$. Smatramo da put u modelu zapravo predstavlja jedan mogući vremenski slijed dinamičkih promjena sustava. Pojedino stanje u nizu odgovara stanju sustava u određenom vremenskom trenutku. Prirodno ćemo, kada je to intuitivno, poistovjećivati stanja u nizu s trenutcima u vremenu. Tako za put $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ možemo reći da se sustav prvo nalazi(o) u stanju s_0, zatim u stanju s_1, itd.. Dio puta π koji počinje od i-tog stanja u nizu označavamo s π^i , npr. $\pi^3 = s_3 \mapsto s_4 \mapsto s_5 \mapsto \cdots$ Kaže se da je π^i **sufiks** puta π.

Ukoliko vrijedi $p \in L(s)$, uobičajeno je i reći da stanje s sadrži p odnosno da čvor s sadrži p. Stanja se ponekad nazivaju čvorovima jer se konačni tranzicijski sustavi (oni koji imaju konačan skup stanja S) mogu prikazati pomoću usmjerenih grafova. Čvorovi takvih grafova odgovaraju stanjima sustava i imaju naznačene propozicionalne varijable koje sadrže. Strelice (usmjereni bridovi) između čvorova označavaju dozvoljene prijelaze između stanja.

Primjer 1.2.3. *Na slici 1 je prikazan jedan tranzicijski sustav koji ima tri stanja:* s_0, s_1 *i* s_2. *Dozvoljeni prijelazi između stanja su* $s_0 \mapsto s_1$, $s_0 \mapsto s_0$, $s_1 \mapsto s_2$ *i* $s_2 \mapsto s_1$. *Stanje* s_0 *sadrži samo propozicionalnu varijablu* r *odnosno vrijedi* $L(s_0) = \{r\}$, *nadalje* $L(s_1) = \{p, q\}$, *te* $L(s_2) = \{q\}$.

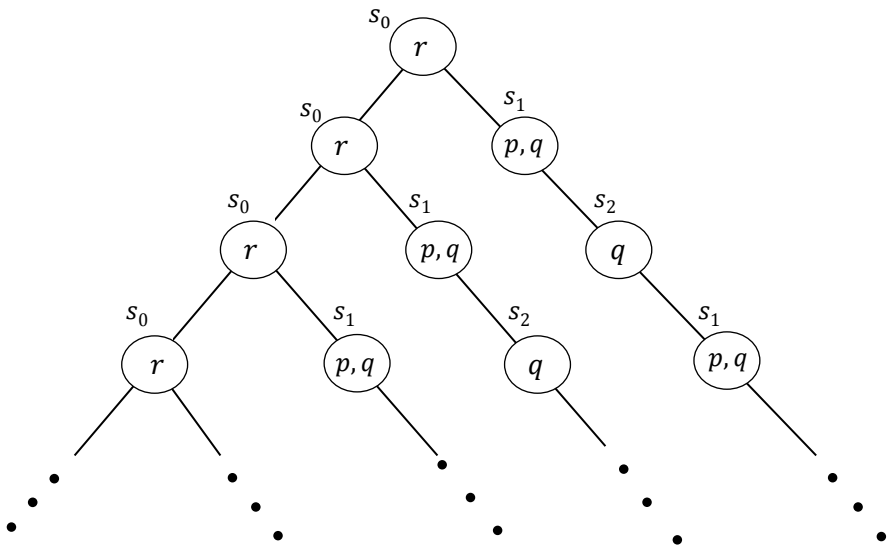


SLIKA 1.1: PRIKAZ TRANZICIJSKOG SUSTAVA POMOĆU USMJERENOG GRAFA

Primjer 1.2.4. *Ponekad je korisno zamisliti sve moguće puteve u modelu iz određenog stanja kao beskonačno stablo. Na slici 1.2, koja se nalazi na sljedećoj stranici, vidimo stablo koje prikazuje sve puteve iz stanja* s_0 *u modelu zadanom slikom 1.1.*

Kao što smo već bili rekli, mi želimo formulama LTL-a opisivati nekakva svojstva tranzicijskih sustava. Ta svojstva će zapravo odgovarati odnosima između stanja koja se pojavljuju na nekom putu, odnosno na nekim putevima. Sada nas zanima kada možemo reći da tranzicijski sustav zadovoljava neka svojstva. U tu svrhu definiramo pojam

istinitosti LTL formule na nekom putu u danom modelu, a zatim i pojam istinitosti formule u stanju modela.



SLIKA 1.2: BESKONAČNO STABLO KOJE PRIKAZUJE SVE PUTEVE IZ STANJA s_0 MODELA ZADANOG SLIKOM 1

Definicija 1.2.5. *Neka je* $\mathcal{M} = (S, \mapsto, L)$ *model i* $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ *put u* $\mathcal{M}$. *Sada rekurzivno definiramo pojam* ***istinitosti*** *LTL formule* ϕ *na putu* π *(pišemo* $\pi \vDash \phi$ *, odnosno* $\pi \nvDash \phi$ *ako ne vrijedi* $\pi \vDash \phi$*) sljedećim pravilima*:

1. $\pi \vDash \top$
2. $\pi \nvDash \bot$
3. $\pi \vDash p \iff p \in L(s_0)$
4. $\pi \vDash \neg\phi \iff \pi \nvDash \phi$
5. $\pi \vDash \phi_1 \wedge \phi_2 \iff \pi \vDash \phi_1 \; i \; \pi \vDash \phi_2$
6. $\pi \vDash \phi_1 \vee \phi_2 \iff \pi \vDash \phi_1 \; ili \; \pi \vDash \phi_2$
7. $\pi \vDash \phi_1 \rightarrow \phi_2 \iff \pi \vDash \phi_2 \; ili \; \pi \nvDash \phi_1$
8. $\pi \vDash X\phi \iff \pi^1 \vDash \phi$
9. $\pi \vDash G\phi \iff \forall i \geq 0, \; \pi^i \vDash \phi$
10. $\pi \vDash F\phi \iff \exists i \geq 0, \; \pi^i \vDash \phi$
11. $\pi \vDash \phi \; U \; \psi \iff \exists n \geq 0 \; t.d. \; za \; 0 \leq i \leq n-1 \; vrijedi \; \pi^i \vDash \phi, \; \pi^n \vDash \psi$
12. $\pi \vDash \phi \; R \; \psi \iff \exists n \geq 0 \; t.d. \; za \; 0 \leq i \leq n \; vrijedi \; \pi^i \vDash \psi, \; \pi^n \vDash \phi;$ $ili \; \pi \vDash G\psi$
13. $\pi \vDash \phi \; W \; \psi \iff \exists n \geq 0 \; t.d. \; za \; 0 \leq i \leq n-1 \; vrijedi \; \pi^i \vDash \phi, \; \pi^n \vDash \psi;$ $ili \; \pi \vDash G\phi$

Pokušajmo sada malo bolje objasniti značenja ovih pravila. Prvo primijetimo da su pravila vezana za logičke veznike analogna pravilima iz logike sudova. Zatim, iz pravila o logičkim veznicima i propozicionalnim varijablama (pravila 3 - 7) vidimo da je istinitost formula LTL-a koje ne sadrže temporalne operatore (tj. "*formula logike sudova*") u potpunosti određena početnim stanjem tog puta. Znači za formule logike sudova možemo reći da se interpretiraju kao u logici sudova, s time da je interpretacija određena vrijednošću funkcije označavanja u početnom stanju puta na kojem se interpretira formula.

Iz pravila o temporalnim operatorima (pravila 8 - 13) vidimo da za njihovu interpretaciju moramo osim početnog, promatrati i druga stanja na putu. Točnije, istinitost formula na putu se određuje promatranjem raznih sufiksa puta. Za interpretaciju operatora *X* (eng. ne**X**t state) promatramo sufiks puta s početkom prvom idućem stanju. Točnije, formula $X\phi$ je istinita na putu π ako i samo ako je formula ϕ istinita na π^1. Npr. ako želimo utvrditi istinitost formule $X(p \vee q)$ na nekom putu $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$, moramo ustanoviti da li je formula $p \vee q$ istinita na sufiksu $\pi^1 = s_1 \mapsto s_2 \mapsto s_3 \mapsto \cdots$. Kako je istinitost formula logike sudova na putu u potpunosti određena početnim stanjem puta, nužno i dovoljno je utvrditi da li je stanje s_1 označeno nekom od propozicionalnih varijabli p ili q. Ako je $\{p, q\} \cap L(s_1) \neq \emptyset$ onda je formula $X(p \vee q)$ istinita na putu π, inače $\pi \nvDash X(p \vee q)$.

Za interpretaciju operatora *F* (eng. some **F**uture state) potrebno je promatrati razne sufikse zadanog puta. Točnije, formula $F\phi$ je istinita na nekom putu π ako i samo ako postoji $i \geq 0$ tako da je formula ϕ istinita na π^i. Formula oblika $F\neg p$ je istinita na nekom putu $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ ako i samo ako postoji neko (buduće), bilo koje stanje tog puta, koje nije označeno varijablom p, tj. ako postoji $i \geq 0$ tako da vrijedi $p \notin L(s_i)$. Primijetimo da iz interpretacije operatora *F* slijedi da pod pojmom budućeg stanja uključujemo i početno stanje, tj. u LTL-u budućnost uključuje i sadašnjost. Znači, ako vrijedi $\pi \vDash \phi$ onda vrijedi i $\pi \vDash F\phi$.

Za interpretaciju operatora *G* (eng. **G**lobally) moramo promatrati globalno sva stanja odabranog puta. Formula oblika $G\phi$ je istinita na nekom putu ako i samo ako je formula ϕ istinita globalno, tj. za svaki $i \geq 0$ formula ϕ je istinita na sufiksu π^i. Npr. formula $G(p \wedge q)$ je istinita na nekom putu π ako i samo ako je svako stanje na tom putu označeno propozicionalnim varijablama p i q. Očito, možemo primijetiti ako vrijedi $\pi \vDash G\phi$ onda vrijedi $\pi \vDash F\phi$ i $\pi \vDash \phi$.

Binarni temporalni operatori *U, W, R* se interpretiraju na slične načine. Ugrubo rečeno, ti operatori zahtijevaju da jedna formula bude istinita sve dok druga to ne postane. Točnije, za operatore *U* (eng. **U**ntil) i *W* (**W**eak-until) formule oblika $\phi\, U\, \psi$ i $\phi\, W\, \psi$ su istinite na nekom putu π ako i samo ako je formula ϕ istinita na svakom sufiksu puta (počevši od nultog), sve dok formula ψ ne postane istinita na nekom sufiksu tog puta. Razlika je u tome što operator *U* zahtijeva da nužno postoji sufiks puta

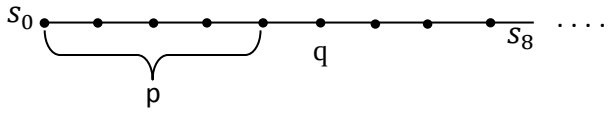
na kojem je formula ψ istinita, dok je za istinitost slabije verzije tog operatora (operatora W) dovoljno da je formula ϕ istinita na svim sufiksima tog puta, tj. da vrijedi $\pi \vDash G\phi$. Npr. formule $p\ U\ q,\ p\ W\ q$ će biti istinite na nekom putu $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ ukoliko postoji $m \in \mathbb{N}$, tako da vrijedi $p \in L(s_i)$, $i \in \{0, \dots, m-1\}$ i $q \in L(s_m)$. No, da formula $p\ W\ q$ bude istinita na putu π dovoljno je i da za svako stanje s na tom putu vrijedi $p \in L(s)$, dok je za istinitost formule $p\ U\ q$ nužno da postoji stanje s na putu za koje vrijedi $q \in L(s)$. Općenitije, formula $\phi\ U\ \psi$ će biti istinita na putu π ako i samo ako postoji $m \geq 0$ tako da je ϕ istinita na π^i, $i \in \{0, \dots, m-1\}$, a ψ je istinita na sufiksu s početkom u stanju s_m, tj. na π^m.

Operator *R* (eng. **R**elease) je sličan operatoru W s time da su zamijenjene uloge argumenata, te se zahtijeva da formula ψ bude istinita sve do i uključujući trenutak kada formula ϕ postane istinita. Formula $p\ R\ q$ će biti istinita na putu π ako i samo ako postoji postoji $m \in \mathbb{N}$ tako da vrijedi $q \in L(s_i), i \in \{0, \dots, m\}$ i $p \in L(s_m)$.

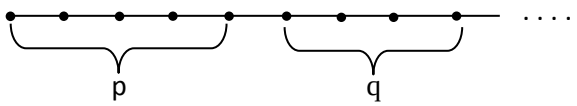
Napomena 1.2.6. *Intuitivno bi iz sugestivnog izraza $'\phi\ Until\ \psi'$ mogli zaključiti da ako je formula $\phi\ U\ \psi$ istinita na nekom putu tada u trenutku kada formula ψ postane istinita onda ϕ nužno mora postati lažna. No, pročitamo li pažljivo pravila iz definicije vidimo da to ne mora vrijediti, tj. ϕ može ostati istinita i nakon što ψ postane istinita. No, dovoljno je da bude istinita sve do trenutka kada formula ψ postane istinita. Također, primijetimo da je za istinitost formule $\phi\ U\ \psi$ dovoljno da formula ψ bude istinita u samo jednom (odgovarajućem) trenutku. Analogna zapažanja vrijede i za operatore W i R. Ilustracije navedenih opaski se mogu vidjeti i u sljedećim primjerima.*

Primjer 1.2.7. *Ovima primjerima na različitim modelima ilustriramo neke specifičnosti interpretacija operatora U, W, R :*

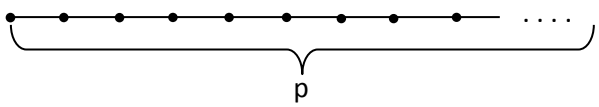
a) Vrijedi $\pi \vDash p\ U\ q,\ \pi \vDash p\ W\ q$ ali $\pi \nvDash q\ R\ p$



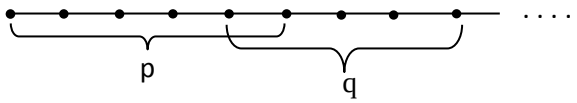
b) Vrijedi $\pi \vDash p\ U\ q,\ \pi \vDash p\ W\ q$ ali $\pi \nvDash q\ R\ p$



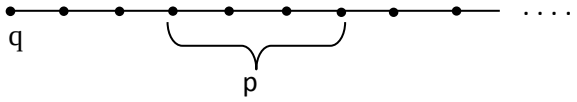
c) Vrijedi $\pi \vDash p\ W\ q$, $\pi \vDash q\ R\ p$ ali $\pi \nvDash p\ U\ q$



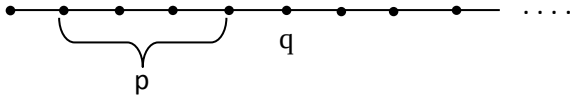
d) Vrijedi $\pi \vDash p\ U\ q, \pi \vDash p\ W\ q,\ \pi \vDash q\ R\ p$



e) Vrijedi $\pi \vDash p\ U\ q,\ \pi \vDash p\ W\ q$, ali $\pi \nvDash q\ R\ p$



f) Vrijedi $\pi \nvDash p\ U\ q, \pi \nvDash p\ W\ q\ ,\ \ \pi \nvDash q\ R\ p$



U realnim situacijama za netrivijalne sustave obično ne možemo u potpunosti predvidjeti budućnost. No, zanimaju nas događaji za koje možemo utvrditi da će se sigurno ostvariti. U LTL-u to znači da je za promatrani tranzicijski sustav i određeno stanje sustava odabrana formula istinita na svim mogućim putevima iz tog stanja.

Definicija 1.2.8. *Neka je* $\mathcal{M} = (S, \mapsto, L)$ *model,* $s \in S$*, i* ϕ *neka LTL formula. Kažemo da je formula* ϕ ***istinita u stanju s*** *(modela* $\mathcal{M}$*), što pišemo* $\mathcal{M}, s \vDash \phi$*, ako za svaki put* π *u* $\mathcal{M}$ *s početkom u* s*, vrijedi* $\pi \vDash \phi$*.*

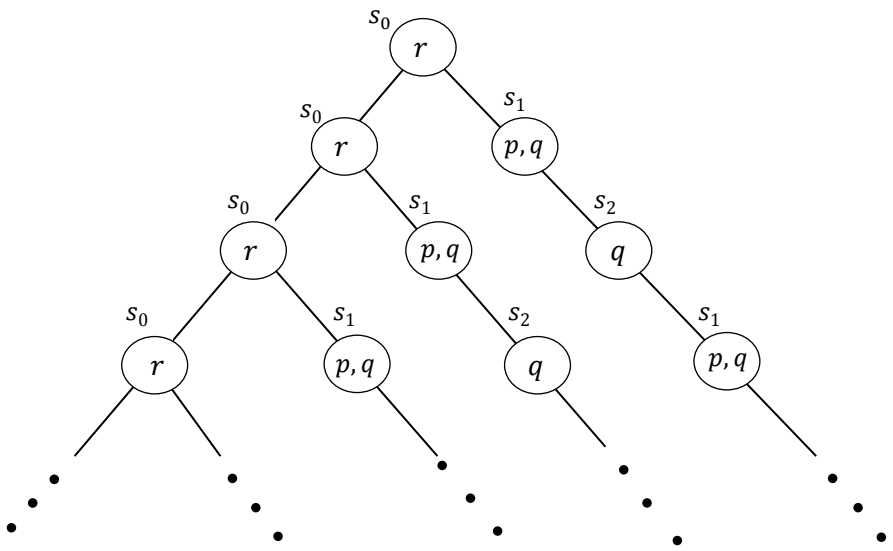
Ako ne vrijedi $\mathcal{M}, s \vDash \phi$ tada pišemo $\mathcal{M}, s \nvDash \phi$ i kažemo da formula ϕ nije istinita u stanju s. Ako je iz konteksta jasno o kojem modelu je riječ onda ćemo umjesto $\mathcal{M}, s \vDash \phi$ skraćeno pisati $s \vDash \phi$.

Istaknimo još jednom da se LTL formule zapravo ne interpretiraju sasvim izolirano u stanjima, tj. osnovna struktura na kojoj se definira istinitost LTL formule je put, i po definiciji formula je istinita u nekom stanju ako i samo ako je istinita na svim putevima iz tog stanja. Specijalno, za formulu logike sudova ϕ i proizvoljni put $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ vrijedi: $\pi \vDash \phi$ ako i samo ako $s_0 \vDash \phi$.

Također, istaknimo da je istinitost u stanju dovoljan uvjet za istinitost na putu, ali ne i nužan. Npr. da bi formula $X\phi$ bila istinita na nekom putu $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$, dovoljno je da formula ϕ bude istinita u stanju s_1. Ali, to nije nužno, tj. nije nužno da formula ϕ bude istinita na svim putevima iz stanja s_1, već samo na putu π^1.

Napomena 1.2.9. *Pojam istinitosti LTL formule možemo definirati i na razini modela, tj. ako je formula* ϕ *istinita u svakom stanju modela* $\mathcal{M}$ *tada ćemo reći da je formula* ϕ ***(globalno) istinita u modelu*** $\mathcal{M}$ *(oznaka* $\mathcal{M} \vDash \phi$*). Za LTL formulu* ϕ *ćemo reći da je* ***valjana*** *ako je istinita u svim modelima* $\mathcal{M}$ *(oznaka* $\vDash \phi$*).*

Primjer 1.2.10. *Kombiniranjem temporalnih operatora možemo izraziti složenija svojstva tranzicijskih sustava, odnosno složenije odnose među stanjima na nekim putevima u modelu. Sljedećim primjerima prikazujemo kako se interpretiraju razne formule u modelima. Primjeri su rađeni nad sustavom sa slike 1.1 čiji je skup svih puteva s početkom u s_0 prikazan na slici 1.2 koju radi praktičnosti ponovno dajemo ovdje.*



1. *Vrijedi $\mathcal{M}, s_0 \vDash r$ jer čvor s_0 sadrži propozicionalnu varijablu r odnosno $r \in L(s_0)$ pa za svaki put π s početkom u s_0 vrijedi $\pi \vDash r$*
2. *Vrijedi $\mathcal{M}, s_0 \vDash \neg p$ jer p nije sadržan u čvoru s_0*
3. *Ne vrijedi $\mathcal{M}, s_0 \vDash Fp$ jer postoji put s početkom u s_0 na kojem niti jedan čvor ne sadrži p. U beskonačnom stablu sa slike 2 to je krajnje lijevi put $s_0 \mapsto s_0 \mapsto s_0 \mapsto \cdots$*
4. *Vrijedi $\mathcal{M}, s_0 \vDash X(p \vee r)$ jer iz stanja s_0 možemo prijeći ili u stanje s_1 koje sadrži p ili opet u stanje s_0 koje pak sadrži r*
5. *Vrijedi $\mathcal{M}, s_0 \vDash G\neg(p \wedge r)$ jer $p \wedge r$ nije istinita u svim stanjima svih puteva s početkom u s_0. Primijetimo $G\phi$ je istinita u modelu za neko stanje s ako je ϕ istinita u svim stanjima do kojih se može doći nekim putem iz s*
6. *Vrijedi $\mathcal{M}, s_0 \vDash F(p \wedge q) \rightarrow FG\neg r$ jer za svaki put π s početkom u s_0 koji sadrži stanje u kojem vrijedi $p \wedge q$, vrijedi $\pi \vDash FG\neg r$. Formula $FG\neg r$ zapravo kaže da nakon nekog stanja u budućnosti nijedan čvor neće sadržavati r. Kako je s_1 jedino stanje u sustavu u kojem vrijedi $p \wedge q$, to znači da svi promatrani putevi prolaze stanjem s_1. Iz slike beskonačnog stabla koje prikazuje sve puteve iz stanja s_0 vidimo da kada put dosegne stanje s_1 on više ne može dosegnuti s_0 koje je jedino stanje sustava u kojem vrijedi r, pa stoga na takvim putevima vrijedi $FG\neg r$.*

7. *Vrijedi* $\mathcal{M}, s_1 \vDash GFp$*, jer iz stanja* s_1 *postoji samo jedan put* $s_1 \mapsto s_2 \mapsto s_1 \mapsto s_2 \mapsto \cdots$*. Formula* GFp *znači da za svako stanje na putu uvijek će postojati stanje u budućnosti koje sadrži* p*, odnosno* p *se pojavljuje beskonačno mnogo puta, što je istina jer* $p \in L(s_1)$*.*
8. *Ne vrijedi* $\mathcal{M}, s_0 \vDash r\ U\ p$*, jer postoji put* $s_0 \mapsto s_0 \mapsto s_0 \mapsto \cdots$ *na kojem* p *nikad neće postati istinit. No, zato vrijedi* $\mathcal{M}, s_0 \vDash r\ W\ p$ *jer sada nije nužno da na svim pu-tevima postoji čvor koji sadrži* p*. Dovoljno je i da svi čvorovi sadrže* r*.*
9. *Ne vrijedi* $\mathcal{M}, s_0 \vDash p\ R\ r$*, jer čvor* s_1 *ne sadrži* r*, a nužno je da na putevima gdje se pojavljuje* p *svi čvorovi sadrže* r *sve do i <u>uključujući</u> stanje koje sadrži* p*.*

Iz definicije istinitosti i navedenih primjera vidimo da se neki operatori slično interpretiraju. U sljedećoj točki želimo istaknuti veze među temporalnim operatorima, a zatim i odrediti adekvatne skupove veznika.

1.3 Važne ekvivalencije formula

Kao što smo vidjeli iz definicija o istinitosti formula LTL-a, interpretacije temporalnih operatora U, W, R se razlikuju u prilično suptilnim detaljima. $Weak - until$ je slabija verzija operatora $Until$, dok se $Release$ interpretira slično kao $Until$ s obrnutim argumentima. U ovoj točki želimo formalno istaknuti neke veze između temporalnih operatora u vidu logički ekvivalentnih formula. Veza između logički ekvivalentnih formula je semantička, odnosno takve formule se interpretiraju na isti način u svim modelima na svim putevima. Formalna definicija logički ekvivalentnih LTL formula slijedi.

Definicija 1.3.1. *Kažemo da su dvije LTL formule* ϕ *i* ψ ***logički ekvivalentne****, ili kratko ekvivalentne, što pišemo* $\phi \equiv \psi$*, ako za sve modele* $\mathcal{M}$ *i sve puteve* π *u* $\mathcal{M}$ *vrijedi:*

$$\pi \vDash \phi \quad \textit{ako i samo ako} \quad \pi \vDash \psi$$

Znači, logički ekvivalentne formule su jednake sa stanovišta semantike i pritom naravno ne moraju imati jednak sintaktički (grafički) zapis. Zamjenom proizvoljne potformule u nekoj formuli χ logički ekvivalentnom formulom dobivamo formulu koja je semantički jednaka početnoj formuli χ. Npr. ako su ϕ i ψ logički ekvivalentne formule onda su i formule $F\phi$ i $F\psi$ logički ekvivalentne.

Za početak istaknimo dualnost između pojedinih parova operatora, tj. operatora čija je negacija ekvivalentna njemu dualnom operatoru s negiranim argumentima. Npr. u logici sudova smo vidjeli da su operatori $\wedge$ i $\vee$ međusobno dualni, tj. negacija konjukcije je ekvivalentna disjunkciji negacija i obratno, negacija disjunkcije je

ekvivalentna konjukciji negacija. Slično, u LTL-u su operatori U i R međusobno dualni, tj. vrijedi:

$$\neg(\phi\ U\ \psi) \equiv \neg\phi\ R\ \neg\psi,\ \neg(\phi\ R\ \psi) \equiv \neg\phi\ U\ \neg\psi.$$

Također F i G su međusobno dualni, a X je dualan sam sebi:

$$\neg F\phi \equiv G\neg\phi,\ \ \neg G\phi \equiv F\neg\phi,\ \neg X\phi \equiv X\neg\phi.$$

Dokazi ovih ekvivalencija se lagano izvode iz definicija. Promotrimo ekvivalenciju $\neg G\phi \equiv F\neg\phi$. Ova ekvivalencija izražava očitu činjenicu koja vrijedi za sve modele, na svim putevima: ϕ ne vrijedi u svim stanjima ako i samo ako postoji stanje u kojem ne vrijedi ϕ. Ekvivalencija $\neg(\phi\ U\ \psi) \equiv \neg\phi\ R\ \neg\psi$ zapravo kaže da na bilo kojem putu proizvoljnog modela, ako ne vrijedi da je ϕ istinita sve dok ψ ne postane istinita tada, ili ψ nikad ne postaje istinita, ili ako ψ postane istinita u nekom trenutku tada u prethodnom trenutku ϕ nužno nije istinita, i obratno.

Primijetimo da jedino operator W nema dulanog operatora. Takav operator se može lagano definirati ali nije od previše koristi pa je zbog toga izostavljen. Formalnu definiciju interpretacije ovog operatora nećemo sada raspisivati, ali možemo reći da bi u skladu s tom definicijom takav operator imalo smisla nazivati $Strong - release$.

Za operator F možemo reći da vrijedi distributivnost prema disjunkciji, a operator G je distributivan prema konjunkciji:

$$F(\phi \vee \psi) \equiv F\phi \vee F\psi,\ \ G(\phi \wedge \psi) \equiv G\phi \wedge G\psi.$$

S druge strane, F nije distributivan prema konjunkciji, a G nije distributivan prema disjunkciji. To znači da postoji model u kojem postoji put na kojem se formule $F(\phi \wedge \psi)$ i $F\phi \wedge F\psi$, odnosno $G(\phi \vee \psi)$ i $G\phi \vee G\psi$ različito interpretiraju. Točnije rečeno, vrijede logičke implikacije ali ne i ekvivalencije, tj. ukoliko je na nekom putu istinita formula $F(\phi \wedge \psi)$ tada je na tom putu istinita i formula $F\phi \wedge F\psi$, ali obrat ne vrijedi. Također, ukoliko je formula $G\phi \vee G\psi$ istinita tada je i formula $G(\phi \vee \psi)$ istinita, ali obrat ne vrijedi. Npr. na putu $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto s_1 \mapsto \cdots$ u modelu prikazanom na slici 1.1 (str. 6) vidimo da vrijedi $\pi \vDash Fq \wedge Fr$ (jer je $q \in L(s_1)$ a $r \in L(s_0)$), ali ne vrijedi $\pi \vDash F(q \wedge r)$ (jer ne postoji stanje na putu π u kojem su q i r istiniti). Također, vidimo da na istom modelu vrijedi $\mathcal{M}, s_0 \vDash G(r \vee q)$ (jer za svako stanje s modela vrijedi $\{r, q\} \cap L(s) \neq \emptyset$), ali $\mathcal{M}, s_0 \nvDash Gr \vee Gq$ (jer postoje stanja modela dostižna iz s_0 u kojima nije istinita varijabla q odnosno r).

Napomena 1.3.2. *Kada kažemo da je stanje s_j dostižno iz stanja s_i tada postoji put u promatranom modelu iz stanja s_i na kojem se nalazi s_j.*

Zasada smo vidjeli koja je veza (dualnost) između dvomjesnih operatora U i R, tj. jednomjesnih operatora F i G. No, postoji veza i između dvomjesnih i jednomjesnih temporalnih operatora, tj. vrijedi:

$$F\phi \equiv \top U\phi, \qquad G\phi \equiv \bot R\phi$$

Iz definicije operatora $Until$ vidimo da ako je na nekom putu formula $\top U\phi$ istinita tada formula ϕ mora postati istinita u nekom trenutku na tom putu tj. mora vrijediti $F\phi$. S druge strane, da bi formula $\top U\phi$ bila istinita dovoljno je da ϕ postane istinita u nekom stanju, jer je logička konstanta $\top$ istinita u svakom stanju, tj. dovoljno je da formula $F\phi$ bude istinita.

Druga ekvivalencija $G\phi \equiv \bot R\phi$ se zapravo može dobiti iz $F\phi \equiv \top U\phi$ negacijom i korištenjem dualnosti između operatora F i G, te U i R. No, promatrajući ekvivalenciju direktno, iz definicije operatora $Release$ vidimo ako je formula $G\phi$ istinita na nekom putu tada je i formula $\bot R\,\phi$ istinita na tom putu. Obratno, ako je $\bot R\,\phi$ istinita tada ϕ mora biti istinita u svim stanjima, jer logička konstanta $\bot$ nije istinita niti u jednom stanju, tj. formula $G\phi$ mora biti istinita.

Veza između operatora $Until$ i njegove 'slabije' verzije $Weak-Until$ je predočena sljedećim ekvivalencijama:

$$\phi\, U\, \psi \equiv \phi\, W\, \psi \wedge F\psi, \tag{1.1}$$

$$\phi\, W\, \psi \equiv \phi\, U\, \psi \vee G\phi. \tag{1.2}$$

Kao što smo već spominjali, jedina razlika pri interpretiranju operatora U i W je u zahtijevu ispunjivosti formule ψ. Za razliku od W, operator U zahtijeva da formula ψ nužno postane istinita u nekom trenutku što postižemo konjukcijom operatora W s $F\psi$. Obratno, operator W možemo izraziti pomoću operatora U ako pri tome uključimo mogućnost da je za ispunjivost dovoljno da formula ϕ bude istinita u svim stanjima.

Također, mogli smo uočiti da postoji sličnost između načina interpretacije operatora R i W. Promatranjem pripadnih definicija interpretacije ovih dvaju operatora vidimo da su pravila interpretacije ista sa zamijenjenim ulogama argumenata. Također, operator R zahtijeva da formula koja je istinita sve do trenutka kada druga formula postane istinita, nužno bude istinita i u tom trenutku. Uočene sličnosti su evidentne iz sljedećih ekvivalencija:

$$\phi\, W\, \psi \equiv \psi\, R\, (\phi \vee \psi), \;\; \phi\, R\, \psi \equiv \psi\, W\, (\phi \wedge \psi) \tag{1.3}$$

Između temporalnih operatora možemo promatrati i jednosmjerne veze, tj. veze kod kojih istinitost jedne formule povlači istinitost druge, ali obratno ne vrijedi.

Definicija 1.3.3. *Reći ćemo da formula* ϕ ***logički slijedi*** *iz formule* ψ*, u oznaci* $\psi \Rightarrow \phi$*, ako za sve modele* $\mathcal{M}$ *i sve puteve* π *u* $\mathcal{M}$ *vrijedi ako je* $\pi \vDash \psi$ *onda* $\pi \vDash \phi$*.*

Očito, vrijedi $\psi \equiv \phi$ ako i samo ako $\psi \Rightarrow \phi$ i $\phi \Rightarrow \psi$. Lako se vidi iz definicija da vrijede sljedeće jednosmjerne logičke implikacije:

$$\phi \, U \, \psi \Rightarrow \phi \, W \, \psi,$$

$$\phi \, R \, \psi \Rightarrow \psi \, W \, \phi,$$

$$G\psi \vee G\phi \Rightarrow \phi \, W \, \psi.$$

1.4 Adekvatni skupovi veznika

U prethodnoj točki smo opisivali razne veze između temporalnih operatora pomoću logički ekvivalentnih formula. Pri tome smo vidjeli da se neki operatori u potpunosti mogu izraziti pomoću drugih. Sada nas zanima postoji li nekakav minimalan podskup temporalnih operatora, tzv. adekvatan skup temporalnih operatora koji ima istu izražajnost kao i definirani skup temporalnih operatora LTL-a. Odgovor je potvrdan. No, postoji više različitih podskupova koji su adekvatni. Slično, u logici sudova možemo pomoću npr. veznika $\neg$ i $\wedge$ izraziti sve preostale logičke veznike. Ipak u alfabete se osim adekvatnih skupova operatora/veznika obično uključuju i neki drugi (redundantni) operatori/veznici koji nam omogućuju lakše zapisivanje nekih inače podužih i nečitkih formula koje se često koriste, a čija interpretacija je obično intuitivna. Prije nego što počnemo s konstrukcijom adekvatnih skupova temporalnih operatora, dajemo formalnu definiciju pojma adekvatnog skupa.

Definicija 1.4.1. *Kažemo da se neki n-mjesni operator* O *može izraziti pomoću skupa operatora* $V = \{V_1, V_2, \dots, V_m\}$ *ako postoji formula F, u kojoj se pojavljuju samo operatori iz skupa V, tako da vrijedi* $O(p_1, p_2, \dots, p_n) \equiv F$*. Kažemo da je skup operatora* V ***adekvatan*** *za skup operatora* W *ako se svaki operator iz* W *može izraziti pomoću operatora iz skupa V i nijedan operator* $f \in V$ *se ne može izraziti pomoću skupa* $V \setminus \{f\}$*.*

Na početku konstrukcije adekvatnog skupa A za LTL uočimo da u prethodnoj točki operator X, za razliku od ostalih operatora nismo izražavali preko drugih operatora. To je zato što je operator X zapravo *ortogonalan* na ostale temporalne operatore, tj. njegovo djelovanje ne možemo izraziti pomoću preostalih operatora. Posebnost operatora X možemo naslutiti iz načina njegove interpretacije. Pomoću operatora X možemo izraziti zahtjev da neka formula mora biti istinita u točno određenom trenutku u odnosu na

početak puta (tj. u prvom sljedećem), dok pomoću ostalih operatora se ne možemo referirati na točno određene trenutke u odnosu na početno stanje. Sada je očito da svaki adekvatan skup temporalnih operatora mora sadržavati operator X, tj. $X \in A$.

Što je s preostalim operatorima? Pa vidjeli smo u prethodnoj točki da se manje više svi mogu izraziti preko drugih. Štoviše, postoje parovi operatora koji su jedan drugom dualni. Vidjeli smo da su operatori U i R međusobno dualni pa stoga sve što možemo izraziti preko jednog operatora možemo izraziti i preko drugog operatora. Dakle, ako jedan od ovih operatora izbacimo iz skupa temporalnih operatora ne gubimo ništa na izražajnosti. Iz ekvivalencija 1.3 (str. 14) u prethodnoj točki vidjeli smo da operator W možemo izraziti preko operatora R (pa onda i U) i obratno. Također, možemo primijetiti da pomoću operatora F i G ne možemo izraziti operatore U, W ili R jer iz definicije interpretacije vidimo da zapravo pomoću F i G ne možemo izraziti zahtjev da neka formula vrijedi do nekog trenutka što zapravo zahtijevaju operatori U, W i R. Sada vidimo da adekvatni skup temporalnih operatora mora sadržavati samo jedan od ovih operatora U, R ili W tj. $card(\{U, R, W\} \cap A) = 1$.

Operatore F i G, kao što smo vidjeli, možemo izraziti preko operatora U i R dok obrat ne vrijedi tako da njihovim isključivanjem ništa ne gubimo na izražajnosti. tj. vrijedi $\{F, G\} \cap A = \emptyset$.

Konačno, kako vrijedi $X \in A$, $\{F, G\} \cap A = \emptyset$ i $card(\{U, R, W\} \cap A) = 1$ slijedi da su sljedeći skupovi temporalnih operatora adekvatni: $\{U, X\}$, $\{R, X\}$, $\{W, X\}$.

1.5 Prošireni LTL

Kao što smo u uvodu napomenuli, LTL je jedna temporalna logika koja modelira vrijeme diskretno, tj. kao linearan niz stanja. Jedna od zanimljivijih specifičnosti ovog modela je izostavljanje koncepta prošlosti. Vrijeme u LTL-u se sastoji isključivo od sadašnjosti i budućnosti. Definirani temporalni operatori odnose se na buduća stanja i ne postoje operatori koji bi nam omogućili referiranje na prošlost, odnosno na prošla stanja. Štoviše, vidjeli smo da je u koncept budućnosti zapravo uključena i sadašnjost, pa bi tako mogli i reći da je definirani model u potpunosti orijentiran na budućnost.

Kao što ćemo vidjeti u drugom poglavlju, pokazuje se da je ovakav model prigodan za praktičnu uporabu (pogotovo u računarstvu). Svojstva, odnosno ponašanje promatranih tranzicijskih sustava opisujemo LTL formulama, čiju istinitost provjeravamo na nizovima stanja i pritom nas ne zanima da li formula vrijedi i u prethodnim stanjima. Odnosno mi želimo opisivati kako će se sustav ponašati a ne kako se je ponašao. U ovom kontekstu, pokazat ćemo da je zadani model bez prošlosti opravdan. Dodavanjem temporalnih operatora prošlosti, koji nam omogućavaju referiranje na prošla stanja, ne dobivamo ništa na izražajnosti jezika.

Napomena 1.5.1. *Radi jednostavnosti, u ovoj točki smatramo da skup temporalnih operatora LTL-a sadrži samo operatore X i U, za koje smo pokazali da čine jedan adekvatan skup temporalnih operatora.*

Operatore X i U nazivat ćemo operatorima budućnosti, a alfabet LTL-a proširujemo operatorima prošlosti Y[5] i S[6]. Prošireni LTL, odnosno tzv. $LTL + PAST$, označavat ćemo skraćeno još i $PLTL$. Definicija formule se mijenja u skladu s dodatakom novih operatora, pri čemu smatramo da je S binarni operator, a Y unarni. Dodatak operatora prošlosti zahtijeva reviziju i definicije istinitosti formula. Sada dajemo rekurzivnu definiciju skupa istinitih formula proširenog LTL-a.

Definicija 1.5.2. *Neka je $\mathcal{M} = (S, \mapsto, L)$ model, $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ put u $\mathcal{M}$ i $t \in \mathbb{N}$. Sada rekurzivno definiramo pojam **istinitosti** PLTL formule ϕ na putu π **u trenutku t** (pišemo $\pi, t \vDash \phi$, odnosno $\pi, t \nvDash \phi$ ako ne vrijedi $\pi, t \vDash \phi$) sljedećim pravilima:*

1. $\pi, t \vDash \top$
2. $\pi, t \nvDash \bot$
3. $\pi, t \vDash p \iff p \in L(s_t)$
4. $\pi, t \vDash \neg\phi \iff \pi, t \nvDash \phi$
5. $\pi, t \vDash \phi_1 \wedge \phi_2 \iff \pi, t \vDash \phi_1 \ i \ \pi, t \vDash \phi_2$
6. $\pi, t \vDash \phi_1 \vee \phi_2 \iff \pi, t \vDash \phi_1 \ ili \ \pi, t \vDash \phi_2$
7. $\pi, t \vDash \phi_1 \rightarrow \phi_2 \iff \pi, t \vDash \phi_2 \ ili \ \pi, t \nvDash \phi_1$
8. $\pi, t \vDash X\phi \iff \pi, t+1 \vDash \phi$
9. $\pi, t \vDash Y\phi \iff t > 0, \ \pi, t-1 \vDash \phi$
10. $\pi, t \vDash \phi \ U \ \psi \iff \exists n \geq t \ t.d. \ za \ t \leq i < n \ vrijedi \ \pi, i \vDash \phi, \ \pi, n \vDash \psi$
11. $\pi, t \vDash \phi \ S \ \psi \iff \exists n \leq t \ t.d. \ za \ n < i \leq t \ vrijedi \ \pi, i \vDash \phi, \ \pi, n \vDash \psi$

Napomena 1.5.3. Iz definicije vidimo da je *tok vremena* modeliran na skupu $\mathbb{N} = \{0,1,2,\ldots\}$, tj. smatramo da početno stanje puta odgovara trenutku $t = 0$, prvo sljedeće stanje odgovara trenutku $t = 1$, itd... Kažemo da je da je trenutak t_1 prethodio trenutku t_2 ako vrijedi $t_1 < t_2$, te smatramo da je za trenutak t prvi idući trenutak $t + 1$, a kada je $t > 0$ tada je prvi prethodni trenutak $t - 1$.

U kontekstu ove proširene definicije pojma istinitosti formula, mogli bismo reći da smo u LTL-u zapravo formule uvijek interpretirali u početnom trenutku. No, definicija istinitosti LTL formula u trenutcima je redundantna. Za LTL formule dovoljno je

[5] eng. **Y**esterday
[6] eng. **S**ince

definirati samo istinitost u početnom trenutku, jer očito za svaku LTL formulu ϕ i proizvoljni model $\mathcal{M}$ vrijedi:

$$(\forall t \in \mathbb{N})(\forall \pi \in \mathcal{M})(\pi, t \vDash \phi \iff \pi^t, 0 \vDash \phi)$$

Drugim riječima, $PLTL$ uistinu predstavlja traženo proširenje LTL-a. Vidimo da LTL zapravo odgovara fragmentu logike $PLTL$ čije formule ne sadrže operatore prošlosti. Formule LTL-a nazivamo još i **formulama budućnosti**, dok **formulama prošlosti** nazivamo formule koje ne sadrže operatore budućnosti. Kao što vidimo, formule logike sudova se smatraju i formulama budućnosti i formulama prošlosti.

Kako sada imamo finiju definiciju istinitosti formula tako profinjujemo i definiciju ekvivalentnih formula.

Definicija 1.5.4. *Kažemo da su formule ϕ i ψ **globalno ekvivalentne,** u oznaci $\phi \equiv_g \psi$, ako za svaki model $\mathcal{M}$ vrijedi:*

$$(\forall t \in \mathbb{N})(\forall \pi \in \mathcal{M})(\pi, t \vDash \phi \iff \pi, t \vDash \psi).$$

*Kažemo da su formule ϕ i ψ **inicijalno ekvivalentne,** u oznaci $\phi \equiv \psi$, ako za svaki model $\mathcal{M}$ vrijedi:*

$$(\forall \pi \in \mathcal{M})(\pi, 0 \vDash \phi \iff \pi, 0 \vDash \psi).$$

Očito, za LTL formule (tj. formule budućnosti) globalna ekvivalencija je istovjetna inicijalnoj, tj. za svake dvije formule LTL-a ϕ i ψ vrijedi: $\phi \equiv_g \psi$ ako i samo ako vrijedi $\phi \equiv \psi$. Općenito, za formule $PLTL$-a globalna ekvivalencija povlači inicijalnu, ali obrat ne mora vrijediti. Npr. formule $Y\top$ i $X\bot$ su inicijalno ekvivalentne, ali nisu globalno ekvivalentne.

1.5.1 Separacija

Separacija je jedan fundamentalan koncept za temporalne logike koji uvodi Gabbay (vidi [6],[7]). Ugrubo, za temporalnu logiku kaže se da ima svojstvo separacije ako za svaku njenu formulu postoji ekvivalentna bulovska kombinacija formula, od kojih svaka ovisi isključivo ili o prošlosti, ili o sadašnjosti, ili o budućnosti. Pokazuje se da $PLTL$ zadovoljava svojstvo separacije, što nam omogućuje opravdavanje izostavljanja operatora prošlosti u LTL-u. Sada navodimo Gabbayev teorem separacije za $PLTL$ iz [18].

Teorem 1.5.5. (*PLTL* separacija) *Za svaku PLTL formulu φ postoji globalno ekvivalentna formula φ_s oblika:*

$$(\psi_1 \rightarrow \mathrm{X}\phi_1) \wedge (\psi_2 \rightarrow \mathrm{X}\phi_2) \wedge \ldots \wedge (\psi_n \rightarrow \mathrm{X}\phi_n), \qquad (1.4)$$

gdje su ψ_i *formule prošlosti, a* ϕ_i *formule budućnosti.*

U orginalnom Gabbayevom teoremu separacije, pomoću Gabbayevog algoritma separacije dobije se malo drugačiji oblik rezultirajuće separirane formule (vidi [6]). Oblik separacije (1.4) lagano se može izvesti iz Gabbayeve separacije. Koristit ćemo ovaj oblik separacije jer nam je pogodniji za proučavanje. Formulu φ_s nazivamo **separacijom od** $\boldsymbol{\varphi}$. Sa $(\psi \rightarrow X\phi)_n$ označavamo separaciju koja se sastoji od n konjukcija. Kažemo da je ψ_i formula prošlosti, a ϕ_i formula budućnosti i-te konjunkcije $\psi_i \rightarrow \mathrm{X}\phi_i$. Korištenjem teorema separacije sada možemo $PLTL$ formule transformirati u inicijalno ekvivalentne LTL formule.

Lema 1.5.6. *Za svaku formulu prošlosti postoji inicijalno ekvivalentna* LTL *formula.*

Dokaz leme lako se provodi indukcijom po broju veznika (logičkih i temporalnih) koji se pojavljuju u formuli, pri čemu (pot)formule oblika $\phi_1 \, S \, \phi_2$ i $Y\phi$ zamjenjujemo s inicijalno ekvivalentnim ϕ_2, odnosno $\perp$.

Teorem 1.5.7. *Za svaku* $PLTL$ *formulu postoji inicijalno ekvivalentna* LTL *formula.*

Dokaz Neka je φ proizvoljna $PLTL$ formula. Prema teoremu separacije znamo da se formula φ može separirati. Neka je $(\psi \rightarrow X\phi)_n$ neka separacija od φ. Iz leme (1.5.6) slijedi da za svaku formulu prošlosti ψ_i postoji inicijalno ekvivalentna formula budućnosti ψ_i^F. Kako je φ globalno pa onda i incijalno ekvivalentna sa $(\psi \rightarrow X\phi)_n$, slijedi da je $\varphi \equiv (\psi^F \rightarrow X\phi)_n$. Očito, $(\psi^F \rightarrow X\phi)_n$ ne sadrži operatore prošlosti, tj. $(\psi^F \rightarrow X\phi)_n$ je LTL formula. ■

Kao što vidimo, u kontekstu inicijalne ispunjivosti, dodavanjem operatora prošlosti ne dobivamo na izražajnosti jezika. Također, korištenjem Gabbayevog algoritma za separaciju (vidi [6]) i prethodno opisane transformacije separiranih formula u inicijalno ekvivalentne LTL formule zapravo dobivamo jedan algoritam za transformaciju $PLTL$ formula u inicijalno ekvivalentne LTL formule. No, korištenjem Gabbayevog algoritma za separaciju dolazi do eksplozije u veličini formule, tj. do neelementarnog[7] povećanja rezultirajuće formule u odnosu na početnu formulu. Za danu $PLTL$ formulu postavlja se pitanje: kolika je veličina najmanje inicijalno ekvivalentne LTL formule?

Točan odgovor, a niti preciznija ocjena ovog tzv. raskoraka u sažetosti[8] između $PLTL$ i LTL formula nije poznata. Osim ovog postoje i druga otvorena pitanja koja su

[7] ELEMENTARY = EXP ∪ 2EXP ∪ 3EXP ∪ ... = DTIME(2^n) ∪ DTIME(2^{2^n}) ∪ DTIME($2^{2^{2^n}}$) ∪ ...
[8] eng. *succinctness gap*

trenutno predmet intenzivnijeg proučavanja. Neka od značajnijih pitanja koja su istaknuta u [10] su:

i) *određivanje složenosti separacije* $PLTL$ *formula*, tj. određivanje složenosti najoptimalnijeg algoritma koji za danu $PLTL$ formulu kao rezultat daje globalno ekvivalentnu separiranu $PLTL$ formulu;

ii) *određivanje raskoraka u sažetosti između* $PLTL$*-a i separiranog* $PLTL$*-a,* tj. za danu $PLTL$ formulu ϕ zanima nas kolika je veličina (u odnosu na formulu ϕ) najmanje globalno ekvivalentne separirane $PLTL$ formule;

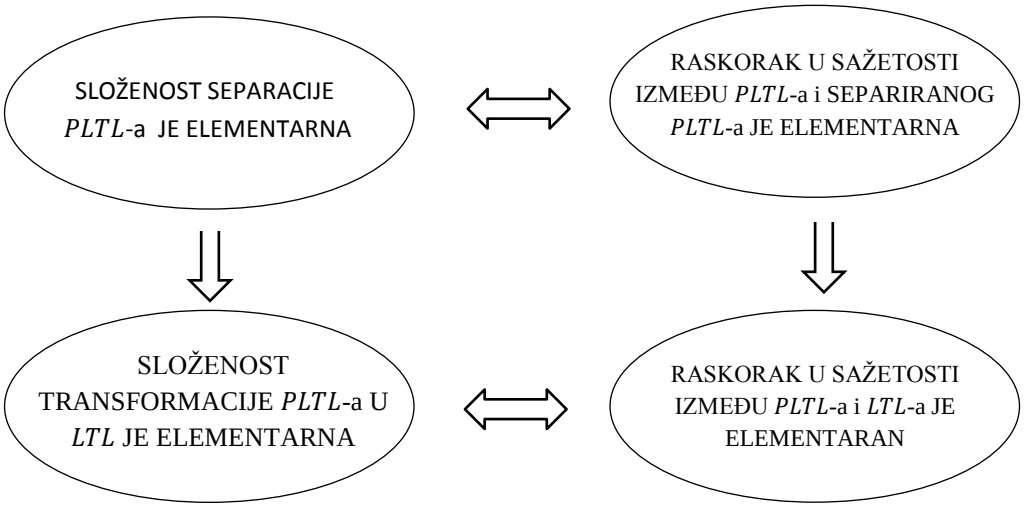
iii) *određivanje složenosti transformacije* $PLTL$*-a u* LTL*,* tj. određivanje složenosti najoptimalnijeg algoritma koji danu $PLTL$ formulu transformira u inicijalno ekvivalentnu LTL formulu.

Očito, možemo primijetiti da između ovih problema postoje neke međuzavisnosti. Vidjeli smo da ako postoji elementaran algoritam za separaciju tada postoji i elementaran algoritam za transformaciju $PLTL$-a u LTL. Također, znamo da ako postoji elementaran algoritam za transformaciju $PLTL$-a u LTL tada je i raskorak u sažetosti između $PLTL$-a i LTL-a elementaran (jer algoritam onda u elementarnom vremenu ispisuje rezultirajuću formulu). No, iz sljedećeg teorema možemo zaključiti da vrijedi i obrat.

Teorem 1.5.8. *Postoji elementaran algoritam za separaciju* $PLTL$*-a ako i samo ako je raskorak u sažetosti između* $PLTL$*-a i separiranog* $PLTL$*-a elementaran.*

Dokaz Jedan smjer je trivijalan. Ako postoji elementaran algoritam za separaciju $PLTL$-a tada se rezultirajuća formula ispisuje u elementarnom vremenu, tj. duljina rezultirajuće separirane formule je elementarna u odnosu na duljinu početne formule. Obratno, ako je raskorak u sažetosti elementaran tada znamo da je duljina tražene formule elementarna u odnosu na ulaznu. Kako je SAT za $PLTL$ elementaran (točnije PSPACE, vidi [23]), te je algoritam za prepoznavanje separiranih formula elementaran (također PSPACE, vidi [10]) tada ispitivanjem svih formula LTL-a (u kojima se pojavljuju iste propozicionalne varijable kao i u početnoj formuli) poredanih po duljini, možemo pronaći traženu separiranu formulu. Očito, opisani algoritam je elementaran. ■

Analogno možemo zaključiti i za odnos složenosti transformacije $PLTL$-a u LTL i raskoraka u sažetosti između $PLTL$-a i LTL-a. Sve navedene međuzavisnosti možemo sumirati sljedećim dijagramom:



Vidimo da se prethodno opisane međuovisnosti zapravo odnose na *gornje* granice. Što možemo reći o *donjim* granicama? Pa, očito vrijedi da ako je raskorak u sažetosti između $PLTL$-a i separiranog $PLTL$-a neelementaran onda je za ispis te formule potrebno neelementarno vrijeme pa je i složenost separacije $PLTL$-a neelementarna. Analogno možemo zaključiti i za odnos složenosti transformacije $PLTL$-a u LTL i raskoraka u sažetosti između $PLTL$-a i LTL-a. Također, možemo primijetiti da ako je složenost transformacije $PLTL$-a u LTL neelementarna tada je i složenost separacije $PLTL$-a neelementarna. Isti odnos vrijedi i za raskorake u sažetosti.

Na kraju, iz prethodno opisanih odnosa u granicama, možemo zaključiti da je raskorak u sažetosti između $PLTL$-a i separiranog LTL-a u najgorem slučaju jednak složenosti separacije $PLTL$-a. Složenost transformacije $PLTL$-a u LTL je manja ili jednaka složenosti separacije $PLTL$-a. Raskorak u sažetosti između $PLTL$-a i LTL-a je u najgorem slučaju jednak složenosti transformacije $PLTL$-a u LTL, odnosno raskoraku u sažetosti između $PLTL$-a i separiranog $PLTL$-a.

Postoje razni rezultati koji opisuju gornje i donje granice složenosti separacije, odnosno transformacije, te veličine raskoraka u sažetosti. Vjeruje se da složenost separacije nije elementarna, dok se za veličinu raskoraka u sažetosti vjeruje da bi mogla biti elementarna. Markey pokazuje da je raskorak u sažetosti između $PLTL$ i LTL formula barem eksponencijalan (vidi [16]).

Petric Maretić, Torabi Dashti i Basin uspostavljaju vezu između problema separacije tzv. *usidrenih* $PLTL$ formula i transformacije $PLTL$ u LTL (vidi [18]). Pokazuje se da postoji elementaran algoritam za separaciju usidrenih formula ako i samo ako postoji elementaran algoritam za transformaciju $PLTL$ u LTL.

Napomena 1.5.9. *Linearnom temporalnom logikom se bave stručnjaci iz različitih područja znanosti (logike, lingvistike, računarstva...) i nažalost ne postoji neka standardna notacija. Pa tako, npr. u [18] LTL zapravo odgovara našem proširenom LTL-u (PLTL), a $FLTL$ se odnosi na fragment bez operatora prošlosti koji zapravo odgovara našem LTL-u.*

1.5.2 Usidreni $PLTL$

Sada ćemo ukratko opisati jedan strogi podskup $PLTL$ formula, tzv. usidrene $PLTL$ formule. Pokazuje se da problem transformacije $PLTL$-a u LTL možemo okarakterizirati pomoću separacije usidrenih $PLTL$ formula. Detaljniju razradu nekih od rezultata koje dajemo u ovoj točki i općenito teorije vezane za usidreni $PLTL$ možete vidjeti u [18].

Neformalno rečeno, usidrene $PLTL$ formule su formule koje imaju istu semantiku u svakom trenutku. Slijedi da je semantika ovakvih formula određena početnim stanjem puta. Za svaku $PLTL$ formulu može se pronaći ekvivalentna usidrena formula pomoću jednostavnih sintaktičkih manipulacija. Prije formalne definicije uvodimo dvije pokrate. Formulu oblika $\top S\phi$ skraćeno zapisujemo $P\phi$, a formulu oblika $\neg P\neg\phi$ (tj. $\bot\ S\ \phi$) s $H\phi$. Možemo uočiti da su za operatore budućnosti G i F, njihovi analogni operatori prošlosti upravo H[9] i P[10].

Definicija 1.5.10. *Kažemo da je $PLTL$ formula ϕ **usidrena** ako je za svaki put π u proizvoljnom modelu $\mathcal{M}$, istnitost formule ϕ jednaka u svim trenutcima. Za $PLTL$ formulu φ, formulu $HP\varphi$ nazivamo **usidrenje od** φ.*

Možemo primijetiti da je usidrenje od φ inicijalno ekvivalentno s φ. Također, primijetimo da je formula $HP\varphi$ istinita u nekom trenutku t ako i samo ako za svaki trenutak j u prošlosti od t $(0 \leq j \leq t)$ postoji njemu prethodni trenutak i $(0 \leq i \leq j)$ u kojem je formula φ istinita. Iz sljedeće leme neposredno slijedi da je usidrenje neke $PLTL$ formule zaista usidrena formula.

Lema 1.5.11. *Za svaku $PLTL$ formulu φ vrijedi:*

(1) $(\forall\mathcal{M})(\forall\pi\in\mathcal{M})(\forall i\in\mathbb{N})(\pi,i\vDash HP\varphi\implies\pi,0\vDash\varphi)$;

(2) $(\forall\mathcal{M})(\forall\pi\in\mathcal{M})(\forall i\in\mathbb{N})(\pi,0\vDash\varphi\implies\pi,i\vDash HP\varphi)$.

Dokaz Neka je π put takav da $\exists i\in\mathbb{N}$ tako da vrijedi $\pi,i\vDash HP\varphi$. Slijedi $\pi,0\vDash P\varphi$, a onda i $\pi,0\vDash\varphi$, čime je dokazana tvrdnja (1).

[9] eng. **H**istory
[10] eng. some **P**ast state

Neka je π put takav da vrijedi $\pi, 0 \vDash \varphi$. Onda, za svaki $j \in \mathbb{N}$ vrijedi $\pi, j \vDash P\varphi$. Posebno, za svaki $i \in \mathbb{N}$ i svaki $j \leq i$ također vrijedi $\pi, j \vDash P\varphi$. Stoga, za svaki $i \in \mathbb{N}$ vrijedi $\pi, i \vDash HP\varphi$, čime je dokazanja tvrdnja (2). ■

Prema teoremu $PLTL$ separacije (1.5.5) znamo da za svaku $PLTL$ formulu postoji globalno ekvivalentna separirana formula oblika $(\psi \to X\phi)_n$. Od posebne zanimljivosti su tzv. kanonske separacije.

Prije definicije kanonske separacije uvodimo jedan novi pojam. Neka je $\mathcal{M} = (S, \mapsto, L)$ proizvoljan model. **Trag** t u modelu $\mathcal{M}$ je konačan niz stanja iz S, u oznaci $t = s_0 s_1 s_2 \dots s_n$, gdje za svaki s_i vrijedi $s_i \mapsto s_{i+1}$. Duljina traga $t = s_0 s_1 s_2 \dots s_n$, u oznaci $|t|$, jednaka je $n + 1$. Konkatenaciju traga t i puta π označavamo s $t\pi$. Za formulu prošlosti ψ i trag t u modelu $\mathcal{M}$, pišemo $t \vDash \psi$ ako $t\pi, |t| - 1 \vDash \psi$, za svaki put $\pi \in \mathcal{M}$. Sada možemo definirati kanonsku separaciju.

Definicija 1.5.12. *(Kanonska separacija) Za PLTL formulu φ,* ***kanonska separacija*** *od φ je separacija $(\psi \to X\phi)_n$ koja zadovoljava sljedeća svojstva:*

(1) za svaki trag t, postoji jedinstveni i tako da vrijedi $t \vDash \psi_i$;
(2) za svaki i, j, $i \neq j$, vrijedi $\phi_i \not\equiv \phi_j$;
(3) postoji i takav da $\phi_i \equiv \bot$;
(4) ako postoji i takav da $\psi_i \equiv_g \bot$, onda vrijedi $\phi_i \equiv \bot$.

Iz prva dva svojstva možemo pročitati da za svako prethodno ponašanje sustava zadano tragom t postoji jedinstvena konjunkcija kanonske separacije čija formula budućnosti određuje dopušteno buduće ponašanje sustava. Pokazuje se da svaku separaciju možemo transformirati u kanonski oblik. Također, iz sljedećeg teorema je vidljivo da sve kanonske separacije neke formule φ imaju istu strukturu. Dokaz teorema može se vidjeti u [18].

Teorem 1.5.13. *Neka su $(\psi \to X\phi)_n$ i $(\psi' \to X\phi')_m$ dvije kanonske separacije neke LTL formule ψ. Tada vrijedi:*

(1) n=m;
(2) za svaki i postoji j tako da vrijedi $\psi_i \equiv_g \psi'_j$ i $\phi_i \equiv \phi'_j$.

Ovaj teorem zapravo opravdava pridjev *kanonski*, u smislu da su za svake dvije kanonske separacije iste formule, odgovarajuće konjunkcije potpuno semantički jednake. Možemo primijetiti da za kanonsku separaciju usidrenja neke formule φ vrijede sva svojstva i iz leme 1.5.11, i iz definicije 1.5.12.

Uspostavljanjem veze između $\omega-$ automata (točnije $Büchijevih\ automata$[11]) i $PLTL-a$, možemo dokazati da je broj konjunkcija kanonske separacije proizvoljne usidrene $PLTL$ formule elementaran i da su veličine formula prošlosti konjunkcija također elementarne. Precizniji rezultati dani su u sljedećim teoremima. Dokazi teorema se mogu vidjeti u [18]. Napomenimo još da se zbog upotrebe $Büchijevih\ automata$ prilikom sljedećih razmatranja u ovoj točki, ograničavamo na konačne modele (konačan broj stanja). U pogledu praktične primjene ovakvo ograničenje nije relevantno.

Teorem 1.5.14. *Za svaku $PLTL$ formulu φ, broj konjunkcija kanonske separacije usidrenja od φ je najviše dvostruko eksponencijalan u veličini formule φ ($\leq 2^{2^{|\varphi|}}$).*

Teorem 1.5.15. *Neka je φ $PLTL$ formula. Postoji kanonska separacija $(\psi \to X\phi)_n$ usidrenja od φ gdje je veličina formula prošlosti svake konjunkcije najviše četverostruko eksponencijalna u veličini formule φ.*

Prije dokazivanja glavnog rezultata dajemo definiciju skupa potformula $PLTL$ formule.

Definicija 1.5.16. *Neka je ϕ proizvoljna $PLTL$ formula. Rekurzivno definiramo* ***skup potformula*** *od ϕ, u oznaci $SF(\phi)$, gdje imamo tri slučaja s obzirom na oblik formule ϕ.*

(1) $SF(*) = \{*\}$ pri čemu je $*\in \{\top, \bot\}$, $SF(p) = \{p\}$ gdje je p propozicionalna varijabla
(2) $SF(\star\, \chi_1) = \{\star\, \chi_1\} \cup SF(\chi_1)$, $\star\, \chi_1$ pri čemu je $\star \in \{\neg, X, Y\}$
(3) $SF(\chi_1 \diamond \chi_2) = \{\chi_1 \diamond \chi_2\} \cup SF(\chi_1) \cup SF(\chi_2)$ pri čemu je $\diamond$ oznaka za dvomjesni logički veznik ili dvomjesni temporalni operator

Sada možemo dokazati glavni rezultat u kojem uspostavljamo vezu između separacije usidrenih $PLTL$ formula i transformacije $PLTL$-a u LTL.

Teorem 1.5.17. *Raskorak u sažetosti između usidrenog $PLTL$-a i separiranog usidrenog $PLTL$-a je elementaran ako i samo ako je raskorak u sažetosti između $PLTL$-a i LTL-a elementaran.*

Dokaz Neka je φ proizvoljna $PLTL$ formula. Tada iz separacije usidrenja od φ konstruiramo LTL formulu φ^F, koja je inicijalno ekvivalentna φ, na način opisan u dokazu teorema 1.5.7. (str. 19). Iz konstrukcije je vidljivo da je formula φ^F očito manja od separacije usidrenja od φ. Stoga, pretpostavljajući da je raskorak u sažetosti između usidrenog $PLTL$-a i separiranog usidrenog $PLTL$-a elementaran, možemo zaključiti i da je raskorak u sažetosti između $PLTL$-a i LTL-a elementaran.

[11] Definicija i svojstava raznih varijanti $\omega-$ automata mogu se naći u [9]

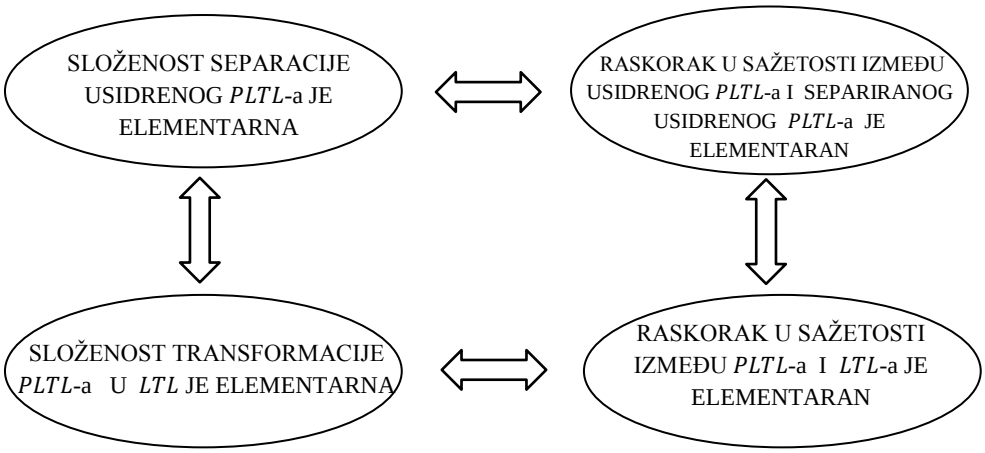
Obratno, neka je φ proizvoljna usidrena $PLTL$ formula i neka je $(\psi \to X\phi)_n$ jedna kanonska separacija od φ. Pretpostavimo da je odnos veličina formule φ i inicijalno ekvivalentne LTL formule φ^F elementaran. Uzmemo li u obzir teoreme 1.5.14. i 1.5.15., vidimo da je za dokaz obrata dovoljno pokazati da je veličina formula budućnosti ϕ_i elementarna u odnosu na $|\varphi^F|$.

Za svaki indeks i i trag $t = s_0 s_1 \dots s_{|t|-1}$ takav da vrijedi $t \vDash \psi_i$ znamo da vrijedi $t\pi, 0 \vDash \varphi^F$ ako i samo ako $\pi, 0 \vDash \phi_i$. Sada, kako je φ^F formula budućnosti, raznim manipulacijama nad operatorima U i X te "parcijalnom evaluacijom" formule φ^F u stanju s_0, možemo transformirati formulu φ^F u formulu oblika $X\varphi_1^F$ (vidi primjer 1.5.18.). Odavde slijedi da vrijedi $t\pi, 0 \vDash \varphi^F$ ako i samo ako $s_1 s_2 \dots s_{|t|-1}\pi, 0 \vDash \varphi_1^F$. Na isti način ponavljamo proceduru za cijeli trag t, dok ne dobijemo formulu $\varphi_{|t|}^F$ za koju vrijedi: $t\pi, 0 \vDash \varphi^F$ ako i samo ako $\pi, 0 \vDash \varphi_{|t|}^F$. Slijedi da je $\varphi_{|t|}^F \equiv \phi_i$. (Također, može se pokazati da se za svaki ϕ_i uvijek može odabrati model $\mathcal{M}$ i trag t u $\mathcal{M}$, čija je duljina najviše dvostruko eksponencijalna u odnosu na $|\varphi|$, takav da vrijedi $t \vDash \psi_i$.)

Da bi dovršili dokaz, trebamo još pokazati da je odnos veličina formula $\varphi_{|t|}^F$ i φ^F elementaran. Iz definicije skupa potformula (1.5.16) slijedi da je broj elemenata skupa $SF(\varphi^F)$ manji ili jednak $|\varphi^F|$. Kako je svaka formula φ_i^F bulovska kombinacija formula iz $SF(\varphi^F)$, specijalno slijedi da je formula $\varphi_{|t|}^F$ bulovska kombinacija potformula od φ^F, pri čemu se svaka potformula pojavljuje najviše $2^{|\varphi^F|}$ puta. ■

Primjer 1.5.18. *U ovom primjeru pokazujemo kako za formulu budućnosti* $\varphi^F \equiv_g p \wedge (q\ U\ r \vee Xp \to r)$ *''parcijalnom evaluacijom'' pronaći formulu* φ_1^F *za koju vrijedi :* $s_0\pi, 0 \vDash \varphi^F$ *ako i samo ako* $\pi, 0 \vDash \varphi_1^F$, $\pi \in \mathcal{M}$ *proizvoljan. Pri čemu za promatrani model* $\mathcal{M}$ *vrijedi* $L(s_0) = \{p, q\}$. *Prvo transformiramo formulu* φ^F *tako da ne postoji potformula oblika* $\phi_1 U\, \phi_2$ *koja nije u dosegu operatora* X. *Kako je* $q\ U\ r \equiv r \vee \big(q \wedge X(q\ U\ r)\big)$, *onda vrijedi* $p \wedge (q\ U\ r \vee (Xp \to r)) \equiv p \wedge \big(r \vee \big(q \wedge X(q\ U\ r)\big) \vee (Xp \to r)\big)$. *Sada možemo "parcijalno evaluirati" dobivenu formulu u stanju* s_0. *Svako pojavljivanje propozicionalne varijable* v, *koje nije u dosegu nekog operatora* X, *zamijenimo s logičkom konstantom* $\top$ *ako je* $v \in L(s_0)$, *odnosno s* $\bot$ *ako* $v \notin L(s_0)$. *Dobivenu formulu sada možemo "skratiti", tj. vrijedi* $\top \wedge \Big(\bot \vee \big(\top \wedge X(q\ U\ r)\big) \vee (Xp \to \bot)\Big) \equiv X(q\ U\ r) \vee \neg Xp$. *Možemo primijetiti da smo "parcijalnom evaluacijom'' u stanju* s_0 *dobili formulu za koju vrijedi :* $s_0\pi, 0 \vDash \varphi^F$ *ako i samo ako* $s_0\pi, 0 \vDash X(q\ U\ r) \vee \neg Xp$. *Sada želimo "izlučiti'' operator* X. *Kako je operator* X *samodualan i distributivan prema disjunkciji, vrijedi* $X(q\ U\ r) \vee \neg Xp \equiv X(q\ U\ r \vee \neg p)$. *Očito, vrijedi* $s_0\pi, 0 \vDash \varphi^F$ *ako i samo ako* $s_0\pi, 1 \vDash q\ U\ r \vee \neg p$, *tj.* $\pi, 0 \vDash q\ U\ r \vee \neg p$. *Znači, tražena formula* φ_1^F *je upravo* $q\ U\ r \vee \neg p$.

Analogon teorema 1.5.8 (str. 20) naravno vrijedi i za usidreni $PLTL$. Stoga, vidimo da iz prethodno dokazanog teorema (1.5.17) slijedi da postoji elementaran algoritam za separaciju usidrenog $PLTL$-a ako i samo ako postoji elementaran algoritam za transformaciju $PLTL$-a u LTL. Sljedećom slikom možemo sumirati dokazane međuzavisnosti problema vezanih za separaciju usidrenog $PLTL$-a, odnosno transformaciju $PLTL$-a u LTL.



Poglavlje 2

Formalna verifikacija

Ispitivanje korektnosti rada računarskih sustava je od velike važnosti za *kritične sustave*, čije greške u radu mogu imati ozbiljne nepoželjne posljedice. Postoje mnogi primjeri gdje su softverske ili hardverske greške računarskih sustava uzrokovale štete ogromnih katastrofalnih razmjera. Stoga je prije puštanja u upotrebu ovakvih sustava vrlo važno utvrditi postojanje eventualnih grešaka u radu. Testiranje je jedna metoda provjere ispravnosti sustava koja može biti neefikasna. Testiranjem se obično provjeravaju samo scenariji za koje očekujemo da bi mogli uzrokovati greške. No, prilikom rada računarskih sustava, pogotovo kada imamo više (a)sinkronih procesa koji se paralelno izvršavaju, može doći do nepredviđenih situacija. U ovom poglavlju želimo pokazati kako korištenjem LTL-a možemo verificirati korektnost računarskih sustava, odnosno programa.

Otkrivanje grešaka u programima s više (a)sinkronih paralelnih procesa (paralelnim programima) je u pravilu teže nego otkrivanje grešaka u sekvencijalnim programima. U većini slučajeva, zbog određenosti redoslijeda izvršavanja operacija, greške u sekvencijalnim programima možemo otkriti uzastopnim testiranjem, dok testiranje paralelnih programa često može biti neefektivno. Zbog nedeterminizma u redoslijedu izvršavanja instrukcija različitih procesa paralelnog programa, pri različitim izvođenjima jednog te istog testa, paralelni program se može izvršavati na različite načine i dati različite rezultate. Greške otkrivene prilikom izvođenja nekog testa ne moraju se pojaviti prilikom ponovnog izvođenja istog testa, ali ne zato što smo ih mi '*ispravili*', nego upravo zbog prethodno opisanog nedeterminizma. Stoga će nam od interesa uglavnom biti formalna verifikacija paralelnih programa.

Ovo poglavlje je podijeljeno u dva dijela: teorijski i praktični. U prvom dijelu razmatramo sintaktičke karakterizacije LTL formula koje izražavaju različita svojstva koja su od interesa u verifikaciji paralelnih programa. Ovakve karakterizacije nam pomažu prilikom odabira prigodne tehnike za dokazivanje korektnosti rada sustava u skladu s promatranim svojstvima. U praktičnom dijelu kroz primjere pokazujemo kako se konkretno LTL upotrebljava prilikom verifikacije paralelnih programa. Proučavat ćemo tzv. metodu provjere modela (eng. model checking), gdje ćemo korektnost sustava utvrđivati ispitivanjem istinitosti odgovarajuće LTL formule u modelu koji odgovara promatranom sustavu.

2.1 Svojstva sustava

Za verifikaciju paralelnih programa najvažnije su dvije vrste svojstava: *svojstva sigurnosti*[1] i *svojstva životnosti*[2]. Ovakvu klasifikaciju svojstava prvi je uveo i proučavao Lamport ([14]). Neformalno, svojstva sigurnosti utvrđuju da se *nešto loše* nikad neće dogoditi, dok svojstva životnosti utvrđuju da će se *nešto dobro* eventualno dogoditi.

Ovakva klasifikacija svojstava nam omogućava da odaberemo najprikladniju metodu za provjeru korektnosti rada, u kontekstu promatranog svojstva. Dodatno, poznato je da sva tzv. *pravedna* izvršavanja programa zadovoljavaju promatrano svojstvo sigurnosti ako i samo ako sva izvršavanja (pravedna ili nepravedna) zadovoljavaju to svojstvo. To znači da se ne moramo brinuti o pravednosti programa prilikom dokazivanja svojstava sigurnosti. S druge strane, pravednost je ključna prilikom dokazivanja svojstava životnosti. Također, formalno dokazivanje korektnosti sustava za određena svojstva ponekad nije prikladno, odnosno moguće. U takvim slučajevima možda je ipak moguće ispitati korektnost sustava testiranjem. Pokazuje se da je svojstva sigurnosti moguće provjeravati i testiranjem. S druge strane, svojstva životnosti nije moguće provjeravati testiranjem. Stoga, u formalno specificiranim sustavima poznavanje vrste svojstva koje neka formula izražava može nam uvelike pomoći prilikom određivanja pristupa ispitivanju korektnosti sustava za promatrano svojstvo. U tu svrhu proučavamo sintaktičke karakterizacije ovih svojstava u LTL-u.

Pnuelli je prvi (1977.g.) predložio upotrebu temporalne logike u specifikaciji i verifikaciji paralelnih programa. Formalno, u kontekstu modela, svojstvo je jednostavno skup puteva. Prvu definiciju svojstva sigurnosti je dao Lamport, dok su općenitiju definiciju, koju ćemo i mi koristiti, dali Alpern i Schneider ([1]). Mi ćemo također proučavati i dvije potklase svojstava sigurnosti: *svojstva sigurnosti s ponavljanjem*[3] i *svojstva jake sigurnosti*[4].

Neformalno, svojstva sigurnosti s ponavljanjem su svojstva sigurnosti koja su zatvorena na uzastopna ponavljanja proizvoljnog stanja na putu. Svojstvo jake sigurnosti je svojstvo sigurnosti s ponavljanjem koje je zatvoreno na izbacivanje stanja. Točnije, za svaki put π iz svojstva jake sigurnosti C vrijedi da izbacivanjem proizvoljan broj stanja s puta π dobivamo opet put iz C. Ako put π pripada nekom svojstvu C, onda još kažemo da put π zadovoljava svojstvo C. Svojstva jake sigurnosti prvi je definirao i karakterizirao Sistla ([22]).

U ovoj točki dajemo sintaktičke karakterizacije svojstava sigurnosti, svojstava sigurnosti s ponavljanjem i svojstava jake sigurnosti. Točnije, pokazat će se da svojstvo

[1] eng. safety properties
[2] eng. liveness properties
[3] eng. safety properties closed under stuttering
[4] eng. strong safety properties

sigurnosti izražavaju sve tzv. pozitivne formule LTL-a, koje od temporalnih operatora sadrže samo operatore W i X, a specijalno, one koje sadrže samo operator W su zatvorene i na ponavljanje. Nadalje, pokazat ćemo da sve pozitivne formule koje sadrže samo operator G izražavaju svojstva jake sigurnosti.

Također ćemo proučavati svojstva životnosti, stabilnosti i *pravednosti*[5]. Neformalno, svojstva životnosti su svojstva za koja vrijedi da se svaki konačan niz stanja može produžiti u put tako da zadovoljava promatrano svojstvo. Svojstva apsolutne životnosti su svojstva životnosti koja su zatvorena na dodavanje prefiksa, tj. za svaki put iz svojstva dodavanjem proizvoljnog prefiksa dobivamo opet put koji zadovoljava isto svojstvo. Svojstva stabilnosti su svojstva koja su zatvorena na sufikse, tj. za svaki put iz svojstva svaki njegov sufiks je također iz tog svojstva. Svojstva pravednosti imaju važnu ulogu u verificiranju svojstava životnosti paralelnih programa. To su zapravo svojstva stabilnosti koja su zatvorena na dodavanje prefiksa. Pokazat ćemo sintaktičke karakterizacije svojstava pravednosti, stabilnosti i apsolutne životnosti u LTL-u. No, karakterizacija životnosti u LTL-u ostaje otvoreno pitanje kao što se navodi u [22].

2.1.1 Notacija

Prije nego počnemo s karakteriziranjem svojstava, uvodimo par konvencija i novih pojmova koji će nam olakšati rad. Za razliku od prvotno postavljenih definicija jezika LTL-a, sada smatramo da je $\{X, W\}$ skup temporalnih operatora, a $\{\neg, \wedge\}$ skup logičkih veznika. Kao što smo već pokazali ovi skupovi su adekvatni i ograničavanjem na njih ne gubimo na izražajnosti jezika. Ostali temporalni operatori, odnosno logički veznici se sada mogu uvesti kao pokrate. Također, smatramo da je skup propozicionalnih varijabli konačan.

U ovoj točki nećemo se ograničavati različitim modelima, tj. proučavanja ćemo u ovoj točki provoditi na najopćenitijem modelu, kojeg označavamo s $\mathcal{M}_{ALL}$. Intuitivno, to je model koji ne postavlja nikakva ograničenja na prijelaze između stanja i za svaki podskup propozicionalnih varijabli postoji stanje koje je označeno svim varijablama (i samo njima) iz tog podskupa. Skup stanja tog modela ćemo označavati sa Σ. Primijetimo, Σ je konačan skup budući da smo pretpostavili da je skup propozicionalnih varijabli konačan. To znači zapravo da u ovoj točki, smatramo da su modeli konačni. Kao što smo i prije konstatirali, ovakvo ograničenje nije relevantno u pogledu praktične primjene, budući da modeli u praksi predstavljaju idealizirane apstrakcije realnih sustava i u pravilu imaju samo konačno mnogo stanja.

Relacija prijelaza modela $\mathcal{M}_{ALL}$ je jednaka Kartezijevom produktu $\Sigma \times \Sigma$. Skup svih (beskonačnih) nizova stanja, odnosno puteva u $\mathcal{M}_{ALL}$, označavamo sa Σ^{ω}. Skup svih konačnih nizova stanja (uključujući i prazan niz ε) označavamo sa Σ^{*}. Svaki element

[5] eng. fairness properties

skupa Σ^* zovemo trag. Vidimo da zapravo na sve modele možemo promatrati kao na podmodele od $\mathcal{M}_{ALL}$. Možemo primijetiti da ako za neku LTL formulu ϕ vrijedi $\mathcal{M}_{ALL} \vDash \phi$ onda vrijedi $\vDash \phi$, i obratno. Sada dajemo definiciju pozitivnih formula koje će nam biti od važnosti prilikom karakterizacije svojstava.

Definicija 2.1.1. *Za LTL formulu u kojoj se nijedan temporalni operator ne nalazi u dosegu veznika* $\neg$ *kažemo da je* ***pozitivna formula****. Za formulu* $\phi \vee \psi$, *pri čemu su* ϕ *i* ψ *pozitivne formule, također kažemo da je pozitivna formula. Za pozitivnu formulu* ϕ *kažemo da je izgrađena samo pomoću temporalnog operatora* $T \in \{W, X, G\}$ *ako se u formuli* ϕ *od temporalnih operatora eventualno pojavljuje samo operator* T.

Ako je $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ neki put, tada s $\pi(i, j)$, $0 \leq i \leq j$, označavamo dio puta koji odgovara nizu $s_i s_{i+1} \dots s_j$. Za $i = j$ vrijedi $\pi(i, j) = s_i$. S $\pi(i)$ označavamo i-ti prefiks puta, tj. $\pi(i) = \pi(0, i)$. Konkatenaciju traga t i puta π označavamo s $t\pi$. Očito, $\forall \pi \in \Sigma^{\omega}, \forall i \geq 0$, vrijedi $\pi(i)\pi^{i+1} = \pi$. Ako su a i b neka stanja tada s $ab\pi$ označavamo konkatenaciju stanja a i b s putom π, tj. put $a \mapsto b \mapsto s_0 \mapsto s_1 \mapsto \cdots$. Ako je a neko stanje, tada s a^+ označavamo neprazni niz u kojem se konačno mnogo puta uzastopno ponavlja stanje a, dok oznaka a^* uključuje i mogućnost da se radi o praznom nizu. S a^{ω} označavamo put na kojem se pojavljuje jedino stanje a. Sada možemo definirati svojstva koja ćemo proučavati u ovoj točki.

2.1.2 Svojstva sigurnosti, životnosti i pravednosti

Dosad smo o svojstvima govorili intuitivno, sada formaliziramo taj pojam u kontekstu modela i LTL-a.

Definicija 2.1.2. ***Svojstvo*** *je podskup od* Σ^{ω}. *Neka je* ϕ *proizvoljna* LTL *formula i neka je* $C = \{\pi \in \Sigma^{\omega} : \pi \vDash \phi\}$. *Tada kažemo da* ϕ ***izražava svojstvo*** C.

Znači, svojstva zapravo formaliziramo kao skupove puteva, a LTL formula izražava svojstvo koje odgovara skupu svih puteva na kojima je istinita. Skup svih puteva na kojima je neka formula ϕ istinita označavamo s $L(\phi)$. Slijedi definicija svojstava sigurnosti.

Definicija 2.1.3. *Za svojstvo* C *kažemo da je* ***svojstvo sigurnosti*** *ako za svaki* $\pi_C \in \Sigma^{\omega}$ *vrijedi:*

$$ako\ (\forall i \geq 0)(\exists \pi \in \Sigma^{\omega})(\pi_C(i)\pi \in C)\ onda\ \pi_C \in C.$$

Znači, ako je C svojstvo sigurnosti i ako se svaki prefiks nekog puta π_C može *produžiti* tako da rezultirajući put pripada C, tada i sam put π_C pripada C. S druge

strane, za svaki $\pi_{\bar{C}} \notin C$ postoji indeks i tako da za svaki $\pi \in \Sigma^{\omega}$ vrijedi $\pi_{\bar{C}}(i)\pi \notin C$. Neformalno, kaže se da za svaki put koji nije iz C postoji nepopravljiv ("lošˮ) prefiks.

Za svojstvo C kažemo da je **zatvoreno na ponavljanje** ako za svaki put $\pi = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_i \mapsto s_{i+1} \mapsto \cdots$ iz C vrijedi da je put $\pi_i = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_i \mapsto s_i \mapsto s_{i+1} \mapsto \cdots$ također iz C. Znači, ako je C zatvoreno na ponavljanje i $\pi \in C$ onda ako uzastopno ponovimo konačno mnogo puta proizvoljno stanje na π dobit ćemo put koji također zadovoljava svojstvo C. Kažemo da je LTL formula zatvorena na ponavljanje ako je $L(\phi)$ zatvoreno na ponavljanje.

Lamport je predlagao drugačiju definiciju svojstava sigurnosti. Prema Lamportu, C je svojstvo sigurnosti ako za svaki put $\pi \in \Sigma^{\omega}$, $\pi = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_i \mapsto s_{i+1} \mapsto \cdots$, vrijedi da je $\pi \in C$ ako i samo ako $\pi(i)s_i^{\omega} \in C, \forall i \geq 0$. Ovakva svojstva ćemo nazivati **svojstva L-sigurnosti**.

Neka je *produženi prefiks od* π put koji dobijemo iz promatranog prefiksa tako da ga *produžimo* ponavljanjem zadnjeg stanja beskonačno mnogo puta. Ako je C svojstvo L-sigurnosti, onda za svaki put $\pi \in \Sigma^{\omega}$ vrijedi da ako je svaki produženi prefiks od π iz C, onda je i π iz C. U [2] je pokazano da ako je svojstvo C zatvoreno na ponavljanje, tada je C svojstvo L-sigurnosti ako i samo ako je C svojstvo sigurnosti. Ovakva svojstva ćemo nazivati **svojstva sigurnosti s ponavljanjem**.

Kažemo da je svojstvo C **zatvoreno na brisanje (izbacivanje)** ako $\forall i > 0, \forall \pi \in C$, $\pi = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_i \mapsto s_{i+1} \mapsto \cdots$, vrijedi $\pi_i \in C$, pri čemu π_i dobijemo iz π brisanjem (izbacivanjem) i-tog stanja (koje nužno ne smije biti početno!), tj. $\pi_i = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_{i-1} \mapsto s_{i+1} \mapsto \cdots$. Kažemo da je formula ϕ zatvorena na brisanje (izbacivanje) ako je svojstvo $L(\phi)$ zatvoreno na brisanje (izbacivanje). Slijedi definicija potklase svojstava sigurnosti koju nazivamo svojstva jake sigurnosti.

Definicija 2.1.4. *Kažemo da je* C ***svojstvo jake sigurnosti*** *ako vrijedi :*

(a) C *je svojstvo sigurnosti s ponavljanjem;*
(b) C *je zatvoreno na brisanje.*

Uvjet (b) zapravo nam govori da ako ne promatramo sustav u određenim trenutcima, promatrano ukupno ponašanje sustava i dalje treba zadovoljavati traženo svojstvo (pretpostavlja se da se početno stanje uvijek promatra).

Svojstvo izraženo formulom oblika $G\phi$, gdje je ϕ formula logike sudova, nazivamo **invarijantno svojstvo**. Kako je istinitost formula logike sudova određena u potpunosti početnim stanjem puta, slijedi da su sva invarijantna svojstva, ujedno i svojstva jake sigurnosti, tj. invarijantna svojstva čine potklasu svojstava jake sigurnosti. Sada definiramo svojstva životnosti.

Definicija 2.1.5. *Kažemo da je* C ***svojstvo životnosti*** *ako vrijedi:*

$$(\forall t \in \Sigma^*)(\exists \pi \in \Sigma^\omega)(t\pi \in C)$$

Znači, ϕ izražava svojstvo životnosti ako i samo ako se svaki trag može produžiti tako da na dobivenom putu formula ϕ bude istinita. Također, možemo primijetiti da je C svojstvo životnosti ako i samo ako vrijedi $\{\pi(i)\colon \pi \in C\} = \Sigma^*$. Intuitivno, C je svojstvo životnosti ako se svaki trag može *popraviti* tako da zadovoljava C.

Kažemo da je C svojstvo **apsolutne životnosti** ako je $C \neq \emptyset$ i $\forall \pi \in C, \forall t \in \Sigma^*$, vrijedi $t\pi \in C$. Primijetimo da je svako svojstvo apsolutne životnosti ujedno i svojstvo životnosti. Kažemo da je C **svojstvo stabilnosti** ako $\forall \pi \in C, \forall i \geq 0$, vrijedi $\pi^i \in C$. Znači, može se reći da su svojstva stabilnosti zatvorena na sufikse.

Lema 2.1.6. *Neka je $C \neq \Sigma^\omega$. Tada je C svojstvo stabilnosti ako i samo ako je komplement $\bar{C} = \Sigma^\omega \backslash C$ svojstvo apsolutne životnosti.*

Dokaz Pretpostavimo da je C svojstvo stabilnosti. Neka je $\pi \in \bar{C}$ i $t \in \Sigma^*$. Ako pretpostavimo da je $t\pi \in C$ onda slijedi $\pi \in C$ (jer je C zatvoren na sufikse), što je kontradikcija jer $\pi \in \bar{C}$. Slijedi, $\forall t \in \Sigma^*$ vrijedi $t\pi \in \bar{C}$, odnosno $\bar{C}$ je svojstvo apsolutne životnosti.

Pretpostavimo da je $\bar{C}$ svojstvo apsolutne životnosti. Neka je $\pi \in C$ i neka je π^i proizvoljan sufiks od π. Ako je $\pi^i \in \bar{C}$ onda iz definicije svojstva apsolutne životnosti slijedi da je $\pi \in \bar{C}$, što je kontradikcija. Slijedi, $\pi^i \in C$, tj. C je zatvoren na sufikse, odnosno C je svojstvo stabilnosti. ■

Još ćemo definirati jednu klasu svojstava koja su značajna prilikom verifikacije sustava.

Definicija 2.1.7. *Kažemo da je C **svojstvo pravednosti** ako je C svojstvo stabilnosti, a ujedno i svojstvo apsolutne životnosti.*

Formulu koja izražava svojstvo pravednosti nazivamo **formulom pravednosti**, formulu koja izražava svojstvo sigurnosti nazivamo **formulom sigurnosti**, itd.

2.1.3 Primjeri svojstava

Sada dajemo primjere svojstava koja smo definirali u prethodnoj točki. Neka su $a, b \in \Sigma$, onda s $ab\Sigma^\omega$ označavamo skup svih puteva koji započinju sa stanjem a, nakon kojeg odmah slijedi stanje b. Možemo primijetiti da je $ab\Sigma^\omega$ jedan primjer svojstva sigurnosti. No, to svojstvo nije zatvoreno na ponavljanje, tj. nije svojstvo sigurnosti s ponavljanjem, jer ponavljanjem prvog stanja dobivamo puteve koji ne zadovoljavaju promatrano svojstvo (npr. $aab^\omega \notin ab\Sigma^\omega$). Skup $a^+b^+\Sigma^\omega$ je svojstvo sigurnosti s

ponavljanjem, ali nije svojstvo jake sigurnosti. Skup $a^*b^*\Sigma^\omega$ je primjer svojstva jake sigurnosti.

Skup $(\Sigma\backslash\{a\})^\omega \cup (\Sigma\backslash\{a\})^* a\Sigma^* b\Sigma^\omega$ je svojstvo životnosti, ali nije svojstvo apsolutne životnosti. Vidimo da se zapravo radi o skupu puteva u kojima se nakon prvog pojavljivanja stanja a obavezno pojavljuje stanje b. Skup $\Sigma^* a\Sigma^\omega$, koji se sastoji od svih puteva na kojima se pojavljuje stanje a barem jedanput, je svojstvo apsolutne životnosti.

Sada dajemo par praktičnih primjera.

Najosnovnije svojstvo sigurnosti za resurse raznih računarskih sustava je da se resursi nikad ne dodjeljuju (allocate) ukoliko ih nitko ne zahtijeva (request). Neka *request*, *alloc* označavaju propozicionalne varijable koje predstavljaju zahtijevanje resursa, odnosno dodjeljivanje resursa. Neka je *REQ* skup svih stanja koja su označena varijablom *request*, a neka je *ALLOC* skup stanja koja su označena varijablom *alloc*. Formalno, opisano svojstvo sigurnosti za resurse koje želimo da sustav zadovoljava (i kojeg ćemo skraćeno zvati *safe*), je skup puteva u kojima svakom pojavljivanju nekog stanja iz skupa *ALLOC* prethodi (ne nužno neposredno) stanje iz skupa *REQ*. Ovaj zahtjev možemo izraziti sljedećom *LTL* formulom (dozvoljavamo da se resursi dodjeljuju u istom stanju kada je i zahtjev napravljen):

$$\neg alloc \; W \; request.$$

Svojstvo sigurnosti za kontinuirano dodjeljivanje resursa (skraćeno *con-safe*), je skup puteva na kojima svakom pojavljivanju nekog stanja iz skupa *ALLOC* prethodi (ne nužno neposredno) stanje iz skupa *REQ* i još dodatno zahtijevamo da između svaka dva pojavljivanja *ALLOC* stanja postoji *REQ* stanje. Ovo svojstvo možemo izraziti sljedećom *LTL* formulom:

$$(\neg alloc \; W \; request) \wedge G\big(alloc \rightarrow X(\neg alloc \; W \; request)\big).$$

Svojstvo ograničene sigurnosti (skraćeno *bounded-safe*), je skup puteva na kojima se nakon svakog pojavljivanja nekog *REQ* stanja, unutar iduća dva stanja, obavezno pojavljuje neko *ALLOC* stanje. Ovo svojstvo izražava sljedeća formula:

$$G\big(request \rightarrow (alloc \vee X alloc \vee XX alloc)\big).$$

Može se pokazati da su *safe*, *con-safe* i *bounded-safe* zaista svojstva sigurnosti. Dodatno, svojstvo *safe* je zatvoreno na ponavljanje, dok *con-safe* i *bounded-safe* to nisu. Znači, *safe* je zapravo svojstvo sigurnosti s ponavljanjem.

Neka je *monotone-p* skup svih puteva koja zadovoljavaju sljedeći uvjet: ili je p zadovoljen u svim stanjima puta, ili ako p nije zadovoljen u nekom stanju puta onda p

nije zadovoljen ni u svim budućim stanjima. Možemo vidjeti da je *monotone-p* svojstvo jake sigurnosti i da se može izraziti sljedećom LTL formulom:

$$G(p \vee G\neg p).$$

Jednostavan primjer invarijantnog svojstva je zahtjev da ne dolazi zastoja u radu (eng. *no-deadlock*). Neka *no-deadlock* označava propozicionalnu varijablu koja predstavlja normalan rad sustava bez zastoja. Formula G*no-deadlock* izražava svojstvo da nikad neće doći do zastoja u radu sustava.

Sada dajemo nekoliko primjera svojstava životnosti. Neka je *live-resource* skup puteva u kojima se nakon svakog pojavljivanja *REQ* stanja eventualno pojavljuje i *ALLOC* stanje. Možemo vidjeti da je *live-resource* svojstvo životnosti, ali nije svojstvo apsolutne životnosti. Primjer pomoću kojeg možemo pokazati da *live-resource* nije svojstvo apsolutne životnosti je proizvoljan put π koji ne sadrži ni *ALLOC,* ni *REQ* stanja. Očito, ovaj put zadovoljava svojstvo *live-resource.* No, put $r\pi$, gdje je r proizvoljno *REC* stanje, ne zadovoljava svojstvo *live-resource.* Svojstvo *live-resource* možemo izraziti sljedećom LTL formulom:

$$G(request \rightarrow F\ alloc).$$

Neka je *terminate* propozicionalna varijabla koja predstavlja završetak izvršavanja programa. Neka je *TERM* skup stanja koja su označena varijablom *terminate*. Skup svih puteva na kojima se pojavljuje proizvoljno stanje iz skupa *TERM* je svojstvo apsolutne životnosti. Ovo svojstvo možemo izraziti formulom $F\ terminate$.

Primjer stabilnog svojstva je invarijantno svojstvo. Sada dajemo primjer svojstva pravednosti. Neka su *send* i *receive* propozicionalne varijable koje predstavljaju slanje i primanje poruka preko nekog kanala. Neka je *SEND* skup svih stanja koja su označena varijablom *send,* a *RECEIVE* skup svih stanja koja su označena varijablom *receive.* Neka je *channel-fairness* skup svih puteva π koji zadovoljavaju sljedeće svojstvo: ako se na putu π pojavljuje beskonačno mnogo stanja skupa *SEND,* onda se na π također pojavljuje i beskonačno mnogo stanja skupa *RECEIVE*. Možemo vidjeti da je svojstvo *channel-fairness* svojstvo pravednosti, koje možemo izraziti sljedećom LTL formulom:

$$GF\ send \rightarrow GF\ receive.$$

2.1.4 Karakterizacija svojstava sigurnosti

U ovoj točki dajemo sintaktičke karakterizacije LTL formula koje izražavaju različite vrste svojstava sigurnosti. Sljedećim teoremom pokazujemo da sve pozitivne formule izražavaju svojstva sigurnosti.

Teorem 2.1.8. *Svaka LTL formula koja ne sadrži temporalne operatore je formula sigurnosti. Ako su* ϕ *i* ψ *formule sigurnosti, onda su i* $\phi \wedge \psi$, $\phi \vee \psi$, $X\phi$, $\phi \, W \, \psi$ *i* $G\phi$ *formule sigurnosti*.

Dokaz Prvo pokazujemo da su sve LTL formule koje ne sadrže temporalne operatore formule sigurnosti. Neka je ϕ proizvoljna LTL formula koja ne sadrži temporalne operatore i neka je $\pi \in \Sigma^{\omega}$. Pretpostavimo da se svaki prefiks puta π može produžiti tako da je formula ϕ istinita na dobivenom putu. Budući da je istinitost formule ϕ u potpunosti određena početnim stanjem puta, slijedi da je onda formula ϕ istinita i na putu π.

Pretpostavimo da ϕ i ψ izražavaju svojstva sigurnosti. Neka je C_{ϕ} svojstvo koje izražava formula ϕ, a C_{ψ} svojstvo koje izražava formula ψ.

Kako je formula $\phi \wedge \psi$ istinita na putu $\pi \in \Sigma^{\omega}$ ako i samo ako vrijedi $\pi \vDash \phi$ i $\pi \vDash \psi$, slijedi $C_{\phi \wedge \psi} = C_{\phi} \cap C_{\psi}$. Neka vrijedi $\pi \notin C_{\phi} \cap C_{\psi}$. Pretpostavimo da za svaki $i \geq 0$, postoji put $\sigma_i \in \Sigma^{\omega}$ tako da je $\pi(i)\sigma_i \in C_{\phi} \cap C_{\psi} \subseteq C_{\phi}, C_{\psi}$. Kako su C_{ϕ} i C_{ψ} svojstva sigurnosti, slijedi $\pi \in C_{\phi}$ i $\pi \in C_{\psi}$. Dakle dobivamo kontradikciju $\pi \in C_{\phi} \cap C_{\psi}$. Znači, za svaki put $\pi \notin C_{\phi} \cap C_{\psi}$ postoji $i \geq 0$, tako da za svaki put $\sigma \in \Sigma^{\omega}$ vrijedi $\pi(i)\sigma \notin C_{\phi} \cap C_{\psi}$. Odavde slijedi da je $C_{\phi} \cap C_{\psi}$ svojstvo sigurnosti, tj. formula $\phi \wedge \psi$ je formula sigurnosti.

Očito, formula $\phi \vee \psi$ izražava svojstvo $C_{\phi} \cup C_{\psi}$. Neka je $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ put takav da za svaki $i \geq 0$, postoji put $\sigma_i \in \Sigma^{\omega}$ tako da je $\pi(i)\sigma_i \in C_{\phi} \cup C_{\psi}$. Pretpostavimo da postoji beskonačno mnogo prefiksa puta π koji se mogu produžiti tako da je dobiveni put iz C_{ϕ} (inače tvrdnja mora vrijediti za C_{ψ}). Neka je $\pi(i)$, $i \geq 0$, prefiks za koji postoji $\sigma_i \in \Sigma^{\omega}$ tako da je $\pi(i)\sigma_i \in C_{\phi}$. Tada za svaki $j \leq i$ vrijedi $\pi(j)\pi(j,i)\sigma_i \in C_{\phi}$. Znači, ako možemo neki prefiks puta π produžiti tako da dobiveni put pripada C_{ϕ}, onda možemo i svaki *kraći* prefiks puta π produžiti tako da dobiveni put pripada C_{ϕ}. Budući da se beskonačno mnogo prefiksa puta π može produžiti tako da je dobiveni put pripada C_{ϕ}, onda se mogu i svi prefiksi puta π produžiti tako da je dobiveni put pripada C_{ϕ}. Kako je C_{ϕ} svojstvo sigurnosti, slijedi $\pi \in C_{\phi} \subseteq C_{\phi} \cup C_{\psi}$. Znači, $C_{\phi} \cup C_{\psi}$ je svojstvo sigurnosti, tj. $\phi \vee \psi$ je formula sigurnosti.

Neka je π put takav da se svaki njegov prefiks može produžiti tako da dobiveni put ispunjava formulu $X\phi$. Tada svaki prefiks puta π^1 možemo produžiti tako da dobiveni put ispunjava formulu ϕ. Kako je ϕ formula sigurnosti, vrijedi $\pi^1 \vDash \phi$ pa onda i $\pi \vDash X\phi$. Znači, $X\phi$ je formula sigurnosti.

Sada ćemo dokazati da formula $\phi \, W \, \psi$ izražava svojstvo sigurnosti. Neka je $\pi \in \Sigma^{\omega}$ takav da $\forall i \geq 0$, $\exists \sigma_i \in \Sigma^{\omega}$ tako da vrijedi $\pi(i)\sigma_i \vDash \phi \, W \, \psi$. Imamo dva slučaja:

1) $\forall i, \forall j,\ 0 \leq i \leq j,\ \exists \sigma_{i,j} \in \Sigma^{\omega}$ tako da vrijedi $\pi(i,j)\sigma_{i,j} \vDash \phi$. Kako ϕ izražava svojstvo sigurnosti slijedi da $\forall i \geq 0$ vrijedi $\pi^{i} \vDash \phi$. Specijalno, vrijedi $\pi \vDash \phi$, a onda i $\pi \vDash \phi\ W\ \psi$.
2) Pretpostavimo da ne vrijedi prvi slučaj, tj. $\exists i, \exists j,\ 0 \leq i \leq j < \infty$ takvi da $\forall \sigma \in \Sigma^{\omega}$ vrijedi $\pi(i,j)\sigma \vDash \neg\phi$. Neka je k najmanji i za koji vrijedi izrečena tvrdnja, te k' najmanji odgovarajući j. Znamo da $\forall i \geq 0, \exists \sigma_i \in \Sigma^{\omega}$ tako da vrijedi $\pi(i)\sigma_i \vDash \phi\ W\ \psi$. Specijalno, za neki $i_0 \geq k',\ \exists \sigma_0 \in \Sigma^{\omega}$ tako da vrijedi $\pi(i_0)\sigma_0 \vDash \phi\ W\ \psi$. Iz pretpostavke slučaja slijedi $\pi(k, i_0)\sigma_0 \vDash \neg\phi$, odnosno $\pi^{k}(i_0)\sigma_0 \vDash \neg\phi$, jer inače bi vrijedilo $\pi(k,k')\pi(k', i_0)\sigma_0 \vDash \phi$ što je kontradikcija s pretpostavkom slučaja. Stoga, iz definicije istinitosti za operator W vidimo da $\exists m \leq k$ tako da vrijedi $\pi^{m}(i_0)\sigma_0 \vDash \psi$ i $\forall l, 0 \leq l < m$, vrijedi $\pi^{l}(i_0)\sigma_0 \vDash \phi$. Budući je $i_0 \geq k'$ proizvoljan slijedi da $\exists m \leq k$ takav da $\forall n \geq m, \exists \sigma_n \in \Sigma^{\omega}$ tako da vrijedi $\pi(m,n)\sigma_n \vDash \psi$ i $\forall l, 0 \leq l < m$, vrijedi $\pi(l,n)\sigma_n \vDash \phi$. Kako su ϕ i ψ formule sigurnosti, slijedi $\pi^{m} \vDash \psi$ i $\forall l, 0 \leq l < m$, vrijedi $\pi^{l} \vDash \phi$, odnosno vrijedi $\pi \vDash \phi\ W\ \psi$.

Sada možemo zaključiti da formula $\phi\ W\ \psi$ izražava svojstvo sigurnosti. Kako vrijedi $G\phi \equiv \phi\ W\ \bot$, onda slijedi da i formula $G\phi$ izražava svojstvo sigurnosti. ■

Primijetimo da su formule iz prethodne točke (2.1.3.) koje izražavaju svojstva *safe*, *con-safe* i *bounded-safe* pozitivne. Svaka formula koja je ekvivalentna nekoj pozitivnoj formuli također izražava svojstvo sigurnosti. Odavde slijedi tvrdnja koju ističemo u sljedećoj napomeni.

Napomena 2.1.9. *Neka je L potpun deduktivni sistem za LTL (vidi [3]). Za LTL formulu ϕ, pišemo $\vdash \phi$ ako je ϕ teorem promatranog sistema. Znači, za svaku LTL formulu vrijedi $\vdash \phi$ ako i samo ako $\vDash \phi$. Neka je SAF skup LTL formula generiran sljedećim pravilima:*

(1) ako je ϕ pozitivna formula onda $\phi \in SAF$,
(2) ako vrijedi $\vdash \phi \leftrightarrow \psi$, $\psi \in SAF$, onda je i $\phi \in SAF$.

Tada je svaki element skupa SAF formula sigurnosti (vidi [22]).

Sljedećim teoremom dajemo karakterizaciju formula koje izražavaju svojstva sigurnosti s ponavljanjem.

Teorem 2.1.10. *Svaka pozitivna formula izgrađena samo pomoću operatora W izražava svojstvo sigurnosti s ponavljanjem.*

Dokaz Neka je ϕ pozitivna LTL formula izgrađena samo pomoću operatora W. Indukcijom po složenosti (broju operatora i veznika koji se pojavljuju) formule ϕ,

dokazujemo da su sve pozitivne formule izgrađene od operatora W zatvorene na ponavljanje. Dokažimo prvo bazu indukcije.

Budući da je istinitost na nekom putu pozitivnih formula u kojima se ne pojavljuju temporalni operatori u potpunosti određena početnim stanjem puta, slijedi da su takve formule zatvorene na ponavljanje. Stoga, formule oblika $p \wedge q$, $p \vee q$ su zatvorene na ponavljanje. Neka je $\pi \in \Sigma^{\omega}$ takav da vrijedi $\pi \vDash p\, W\, q$, za neke propozicionalne varijable p, q. Imamo dva slučaja:

1) $\pi^i \nvDash q, \forall i \geq 0$
   Tada vrijedi $\pi^j \vDash p, \forall j \geq 0$. Budući je istinitost propozicionalne varijable p na nekom putu u potpunosti određena početnim stanjem puta, slijedi $p \in L(s)$ za svako stanje s na putu π. Stoga, ponavljanjem proizvoljnog stanja na putu π, dobivamo put π_{pon} za koji opet vrijedi $p \in L(s)$, za svako stanje s koje se pojavljuje na putu π_{pon}. Slijedi $\pi_{pon}^j \vDash p, \forall j \geq 0$, a onda i $\pi_{pon} \vDash p\, W\, q$.
2) $\exists m \geq 0,\ \pi^m \vDash q$ i $\forall i, 0 \leq i < m,\ \pi^i \vDash p$
   Neka je π_{pon} put koji dobijemo iz puta $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ ponavljanjem stanja $s_j, j \geq 0$, tj. $\pi_{pon} = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_{j-1} \mapsto s_j \mapsto s_j \mapsto s_{j+1} \mapsto \cdots$. Kako iz pretpostavke slučaja vrijedi $p \in L(s_i), 0 \leq i < m$, i $q \in L(s_m)$ i budući je istinitost propozicionalne varijable p na putu u potpunosti određena početnim stanjem tog puta, tada u slučaju da je $j < m$ vrijedi $\pi_{pon}^i \vDash p$, $0 \leq i < m$+1 i $\pi_{pon}^{m+1} \vDash q$. U slučaju da je $j \geq m$, vrijedi $\pi_{pon}^i \vDash p$, $0 \leq i < m$ i $\pi_{pon}^m \vDash q$.

Pretpostavimo da tvrdnja vrijedi kada je složenost formule ϕ manja ili jednaka nekom $n \in \mathbb{N}, n \geq 1$. Sada dokazujemo tvrdnju teorema kada je složenost formule ϕ jednaka $n+1$. Za formule oblika $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, pri čemu je složenost formula ϕ_1 i ϕ_2 manja od $n+1$, tvrdnja slijedi direktno iz pretpostavki. Ako je $\phi \equiv \phi_1\, W\, \phi_2$, dokaz provodimo analogno kao i za formulu $p\, W\, q$, pri čemu sada iskorištavamo pretpostavku indukcije da tvrdnja vrijedi za formule ϕ_1 i ϕ_2.

Znači, sve pozitivne formule izgrađene od operatora W su zatvorene na ponavljanje. Iz teorema 2.1.8. slijedi da su to ujedno i formule sigurnosti. ■

Primijetimo da je formula iz prethodne točke koja izražava svojstvo *safe* upravo ovakvog oblika. Svaka formula koja je ekvivalentna s nekom pozitivnom formulom izgrađenom od operatora W također izražava svojstvo sigurnosti s ponavljanjem.

Napomena 2.1.11. *Neka je $SAAF$ skup LTL formula generiran sljedećim pravilima:*

(1) ako je ϕ pozitivna formula izgrađena samo pomoću operatora W, tada $\phi \in SAAF$,
(2) *ako vrijedi $\vdash \phi \leftrightarrow \psi$ (teorem sistema L iz napomene 2.1.9.), $\psi \in SAAF$, tada je $\phi \in SAAF$.*

Tada je svaki element skupa $SAAF$ formula sigurnosti s ponavljanjem (vidi [22]).

Sada želimo karakterizirati svojstava jake sigurnosti. Ponovimo, svojstva jake sigurnosti su svojstva sigurnosti s ponavljanjem koja su zatvorena na brisanje stanja.

Lema 2.1.12. *Svaka pozitivna formula izgrađena samo pomoću operatora G izražava svojstvo jake sigurnosti.*

Dokaz Neka je ϕ proizvoljna pozitivna formula u kojoj se od temporalnih operatora pojavljuje eventualno samo operator G. Budući da je formula oblika $G\psi$ zapravo pokrata za formulu $\psi\, W\, \bot$, iz teorema 2.1.10. slijedi da ϕ izražava svojstvo sigurnosti s ponavljanjem. Znači, potrebno je još dokazati da za svaki put π takav da je $\pi \vDash \phi$ vrijedi: ako izbrišemo bilo koje stanje osim početnog s puta π, tada dobiveni put također ispunjava ϕ. Ovu tvrdnju možemo dokazati indukcijom po složenosti strukture formule ϕ. Ako ϕ ne sadrži temporalne operatore, tada brisanjem stanja s puta π ne utječemo na istinitost formule ϕ, budući da je istinitost formule ϕ u potpunosti određena početnim stanjem puta π. Stoga promatramo samo formule u kojima se pojavljuje operator G.

Očito, formula oblika Gp je zatvorena na brisanje. U koraku indukcije, lagano se pokaže da su formule oblika $\phi \wedge \psi$ i $\phi \vee \psi$ zatvorene na brisanje stanja, budući da za formule ϕ i ψ vrijedi tvrdnja indukcije po pretpostavci. Neka je ϕ formula oblika $G\psi$ i neka je $\pi = s_0 \mapsto s_1 \mapsto s_2 \mapsto \cdots$ put takav da vrijedi $\pi \vDash G\psi$. Tada $\forall i \geq 0$ vrijedi $\pi^i \vDash \psi$. Neka je π_{bris} put koji dobijemo brisanjem stanja s_j, $j > 0$, s puta π, tj. $\pi_{bris} = s_0 \mapsto s_1 \mapsto \cdots \mapsto s_{j-2} \mapsto s_{j-1} \mapsto s_{j+1} \mapsto s_{j+2} \mapsto \cdots$. Vrijedi $\pi^i_{bris} = \pi^{i+1}$, $i > j$, a budući da je po pretpostavci formula ψ zatvorena na brisanje vrijedi i $\pi^i_{bris} \vDash \psi, 0 \leq i < j$. Slijedi $\pi^i_{bris} \vDash \psi, \forall i \geq 0$, tj. $\pi_{bris} \vDash G\psi$. ■

Ako ponovno pogledamo formulu koja izražava svojstvo *monotone-p (str. 33)*, možemo vidjeti da je izgrađena samo pomoću operatora G. Uspostavljanjem veze između Büchijevih automata i LTL-a, pokazat ćemo da vrijedi i obrat tvrdnje iz prethodne leme (2.1.12.). Kao posljedicu dobivamo da klasa formula jake sigurnosti točno odgovara klasi pozitivnih formula izgrađenih samo pomoću operatora G. Büchijevi automati su varijanta konačnih automata koji *čitaju* beskonačne nizove ulaznih simbola (tzv. $\omega-$automata). Sada definiramo Büchijeve automate. Detaljnija obrada teorije vezane za konačne automate, $\omega-$automate i Büchijeve automate može se pronaći u [9], [17] i [25].

Definicija 2.1.13. ***Büchijev automat*** *je uređena petorka $A = (\Delta, Q, \delta, q_0, F)$, gdje je Δ proizvoljan neprazan skup koji nazivamo alfabet automata A, Q je proizvoljan konačan skup čije elemente nazivamo stanjima, δ je funkcija definirana na $Q \times \Delta$, q_0*

nazivamo početno stanje, a $F \subseteq Q$ *nazivamo skup završnih stanja, odnosno skup stanja prihvaćanja. Funkciju* δ *nazivamo funkcijom prijelaza i s obzirom na njezinu kodomenu razlikujemo dvije različite vrste Büchijevih automata. Ako je* $\delta: Q \times \Delta \longrightarrow Q$, *tada kažemo da je* A ***deterministički Büchijev automat (DBA)***. *Ako je* $\delta: Q \times \Delta \longrightarrow 2^Q$, *tada kažemo da je* A ***nedeterministički Büchijev automat (NBA)***.

Prilikom definicije Büchijevih automata možemo smatrati i da automat umjesto jednog početnog stanja sadrži više početnih stanja. Skup početnih stanja tada označavamo s I. Neka je $A = (\Delta, Q, \delta, I, F)$ NBA, $t \in \Delta^\omega$. Beskonačni niz stanja $q_0, q_1, q_2, \ldots$ takav da je $q_0 \in I$ i $\forall i \geq 0$, $q_{i+1} \in \delta(q_i, t_i)$, nazivamo **grana izvršavanja** automata A za ulazni niz t. Kažemo da A **prihvaća** t ako postoji grana izvršavanja od A za t u kojoj se pojavljuje neko završno stanje beskonačno mnogo puta. Za naše potrebe, u ovoj točki promatramo samo automate čiji je alfabet jednak skupu Σ.

Büchijev automat je zapravo jedna varijanta automata koji prihvaćaju beskonačne riječi ($\omega-$automata). **Konačni automati** su automati koji prihvaćaju konačne riječi. Jedina razlika između konačnih automata i Büchijevih automata je zapravo u načinu prihvaćanja ulaznog niza simbola. Konačni automati su znači iste uređene petorke kao i Büchijevi automati, pri čemu konačni automati *čitaju* konačne nizove simbola i prihvaćaju neku riječ $t \in \Delta^*$ ako prilikom *čitanja* t postoji grana izvršavanja koja završava u nekom stanju prihvaćanja. Analogno kao i kod Büchijevih automata, razlikujemo determinističke i nedeterminističke konačne automate. Pokazuje se da je klasa tzv. regularnih jezika jednaka klasi jezika koje prihvaćaju konačni automati (vidi [20]), dok je klasa tzv. $\omega-$regularnih jezika jednaka klasi jezika koje prihvaćaju nedeterministički Büchijevi automati (vidi [9]).

Za niz (stanja automata, ulaznih simbola, vrhova grafa...) $q = q_0, q_1, q_2, \ldots$ s $q(i)$, $i \geq 0$ označavamo konačni niz $q_0, q_1, \ldots q_i$, kojeg nazivamo i-ti prefiks niza q. S $q(i,j), 0 \leq i \leq j$, označavamo konačni niz stanja $q_i, q_{i+1}, \ldots q_j$, a s q^i, $i \geq 0$, označavamo niz $q_i, q_{i+1}, q_{i+2}, \ldots$ kojeg nazivamo i-ti sufiks niza q. Duljinu niza p označavamo s $|p|$. Za beskonačni niz q, pišemo $|q| = \infty$, a duljina i-tog prefiksa niza q je $|q(i)| = i + 1$. Kažemo da je p **podniz** niza q ako postoji niz $0 \leq i_0 < i_1 < i_2 \ldots$ tako da je $p_j = q_{i_j}, \forall j, 0 \leq j < |p|$.

Definicija 2.1.14. *Kažemo da NBA* A *definira svojstvo* C *ako* $\forall \pi \in \Sigma^\omega$ *vrijedi:*

$$\pi \in C \text{ ako i samo ako } A \text{ prihvaća } \pi.$$

Sa $L(A)$ *označavamo svojstvo koje definira automat* A.

Sada želimo pokazati da svako svojstvo jake sigurnosti C definirano nekim NBA A možemo izraziti pozitivnom formulom koja je izgrađena samo pomoću operatora G.

Lema 2.1.15. *Ako je C svojstvo jake sigurnosti definirano nekim NBA A, tada postoji pozitivna formula izgrađena samo pomoću operatora G koja izražava C.*

Dokaz Neka je $A = (\Sigma, Q, \delta, I, F)$ NBA koji definira C. U skladu s A definiramo usmjereni graf $\Gamma = (V, E)$ tako da vrijedi: $V = (Q \times \Sigma) \cup \{v_0\}$ i $E = \{((q,a),(q',a')) : q' \in \delta(q,a')\} \cup \{(v_0,(q',a')) : q' \in \delta(q,a'),\ za\ neki\ q \in I\}$. Primijetimo da ne postoje bridovi koji su usmjereni prema vrhu v_0. Svaki vrh iz skupa V odgovara jednom stanju NBA A i elementu alfabeta Σ. Svaki brid iz skupa E odgovara tranziciji kojom automat A prelazi iz stanja u stanje prilikom čitanja ulaznog simbola iz alfabeta Σ. Neka je $p = p_0, p_1, p_2 \ldots$ beskonačan niz vrhova grafa Γ, pri čemu je $p_0 = v_0$ i $\forall i \geq 0,\ p_i = (q_i, a_i)$. Za promatrani niz vrhova p definiramo pripadni niz ulaznih simbola $t(p) = t_0, t_1, t_2 \ldots$, pri čemu vrijedi $\forall i \geq 0,\ p_{i+1} = (q_{i+1}, t_i)$. Niz p je put ako $\forall i \geq 0,\ (p_i, p_{i+1}) \in E$. Kažemo da je put p prihvatljiv ako postoji beskonačno mnogo indeksa $i \in \mathbb{N}$, tako da je $q_i \in F$. Neka je $L(G) = \{t(p) \in \Sigma^\omega : p$ je prihvatljiv put u G s početkom u $v_0\}$. Iz konstrukcije grafa Γ i definicija skupova $L(A)$ i $L(G)$, vidimo da vrijedi $L(A) = L(G)$. Ako je C prazan skup onda bilo koja antitautologija logike sudova izražava C. Stoga, pretpostavimo da C nije prazan skup.

Neka je V' skup svih vrhova grafa Γ takvih da vrijedi: $v \in V'$ ako i samo ako postoji prihvatljiv put u Γ, s početkom u v_0, koji prolazi stanjem v. Očito, $v_0 \in V'$ budući da svi prihvatljivi putevi započinju u v_0. Neka je $\Gamma' = (V', E')$ gdje je $E' = \{(r,s) : r, s \in V', (r,s) \in E\}$. Znači, Γ' odgovara restrikciji grafa Γ na skup vrhova V'. Ako je p neki prihvatljiv put u grafu Γ, tada po definiciji svi čvorovi tog puta su sadržani i u skupu V', a svi bridovi duž tog puta su sadržani i u E'. Slijedi da je svaki prihvatljiv put u Γ ujedno i prihvatljiv put u Γ'. Kako je Γ' podgraf grafa Γ, tada je svaki prihvatljiv put u Γ' ujedno prihvatljiv i u Γ. Znači, zapravo smo dokazali da vrijedi $L(\Gamma') = L(\Gamma) = L(A) = C$.

Neka je $p = p_0, p_1, p_2 \ldots$ proizvoljan put u grafu Γ' s početkom u v_0. Iz definicije grafa Γ' znamo da za svaki vrh $v \in V'$ postoji prihvatljiv put u Γ' koji prolazi vrhom v. Posebno, za proizvoljan vrh p_i koji se pojavljuje na putu p postoji $j \in \mathbb{N}$ i prihvatljivi put $r = r_0, r_1, r_2 \ldots$ u Γ' tako da je $r_j = p_i$. Budući da se na putu r beskonačno mnogo puta pojavljuju vrhovi čija su pripadna stanja prihvatljiva, tada se i na putu $r^j = r_j, r_{j+1}, r_{j+2} \ldots$ beskonačno mnogo puta pojavljuju vrhovi čija su pripadna stanja prihvatljiva. Sada možemo spojiti prefiks puta p i sufiks puta r tako da dobijemo put $p_0, p_1, \ldots, p_{i-1}, p_i, r_{j+1}, r_{j+2} \ldots$ u Γ' koji je prihvatljiv. Znači, dokazali smo da za put $p = p_0, p_1, p_2 \ldots$ u Γ' s početkom u v_0 vrijedi: za svaki prefiks $p(i) = p_0, p_1, \ldots, p_i$ puta p postoji put r u Γ' tako da je put $p(i)r$ prihvatljiv u Γ'. Budući da je $L(\Gamma') = C$ svojstvo sigurnosti to znači da je put p u Γ' prihvatljiv. Kako

je p proizvoljan put u grafu Γ' s početkom u v_0, dokazali smo da vrijedi sljedeća tvrdnja.

Tvrdnja 1. $C = \{t(p) : p \text{ je put u } \Gamma' \text{ koji počinje u vrhu } v_0\}$

Tvrdnja 2. *Neka je* $p = p_0, p_1, p_2 \ldots$ *proizvoljan niz vrhova grafa* Γ'*, takvih da vrijedi* $p_0 = v_0$, $(v_0, p_1) \in E'$ *i svaki prefiks od* p *je podniz nekog puta u* Γ' *koji počinje u* v_0*. Tada vrijedi* $t(p) \in C$.

Sada dokazujemo tvrdnju 2. Neka je p niz koji ima navedena svojstva. Budući da je svaki prefiks od p podniz nekog puta r u Γ' koji počinje u v_0, tada je svaki prefiks od $t(p)$ podniz od $t(r)$, pri čemu je r neki put u Γ' koji počinje u v_0. Kako je C svojstvo jake sigurnosti, tj. zatvoreno je na brisanje stanja, slijedi da je svaki prefiks od $t(p)$ ujedno i prefiks nekog puta u C. Budući da je C svojstvo sigurnosti, slijedi da je $t(p) \in C$ čime je dokazana tvrdnja 2.

Neka je $\Gamma'' = (V'', E'')$ usmjereni graf gdje je V'' skup *komponenti jake povezanosti*[6] grafa Γ' i $(C_1, C_2) \in E''$ ako i samo ako postoji brid u Γ' iz nekog vrha komponente C_1 u neki vrh komponente C_2. Budući da vrh v_0 ne leži ni na jednom ciklusu, slijedi da je skup $\{v_0\}$ komponenta jake povezanosti grafa Γ' i vrijedi $\{v_0\} \in V''$. Graf Γ'' je acikličan, inače ako bi postojale dvije komponente jake povezanosti C_1 i C_2 koje bi ležale na ciklusu u Γ'', tada bi svaka dva vrha iz C_1 i C_2 ležala na nekom ciklusu u grafu Γ'. U tom slučaju bi skup vrhova $C_1 \cup C_2$ bio jako povezan, što je kontradikcija budući da su C_1 i C_2 maksimalni jako povezani podskupovi vrhova u Γ'. Znači, nijedan put u Γ'' ne sadrži dva jednaka vrha. Budući da je broj vrhova grafa Γ'' konačan, ukupan broj puteva u grafu Γ'' je također konačan.

Neka je $v = (q, a) \in V' \backslash \{v_0\}$. Tada definiramo da je $g(v) = g_1 \wedge g_2$, pri čemu je g_1 konjunkcija svih propozicionalnih varijabli kojima je označeno stanje $a \in \Sigma$, a g_2 je konjunkcija negacija svih ostalih propozicionalnih varijabli. Za svaki $C \in V''$, neka je $g(C)$ disjunkcija svih $g(v)$ takvih da je $v \in C$. Neka je $d = C_0, C_1, \ldots, C_m$ put u Γ''. Definiramo LTL formule $h_m, h_{m-1}, \ldots, h_0$ rekurzivno:

- $h_m = G\big(g(C_m)\big)$,
- $h_i = G(g(C_i) \vee h_{i+1}),\ 0 \leq i < m$.

Pretpostavimo da je $C_0 \neq \{v_0\}$ i postoji brid u Γ' iz v_0 u barem jedan vrh iz C_0. Ovakav put nazivamo *obveznim*. Neka je D skup svih vrhova $v \in C_0$ takvih da postoji

[6] Komponenta jake povezanosti usmjerenog grafa je, ili maksimalni skup vrhova takvih da svaka dva vrha iz tog skupa leže na nekom ciklusu, ili je to jednočlani skup s vrhom koji pokazuje na samog sebe i ne leži ni na jednom drugom ciklusu, ili jednočlani skup s vrhom koji ne leži ni na jednom ciklusu.

brid između v_0 i v. Neka je $g(d) = g(D) \wedge h_0$. Neka je ϕ disjunkcija svih $g(d)$ takvih da je d obvezan put u Γ''.

Tvrdnja 3. *Za svaki* $\pi \in \Sigma^\omega$ *vrijedi:* $\pi \vDash \phi$ *ako i samo ako* $\pi \in C$.

Dokažimo tvrdnju 3. Pretpostavimo da je $\pi \in C$. Iz tvrdnje 1. slijedi da postoji put p u grafu Γ' s početkom u v_0 takav da je $\pi = t(p)$. Iz konstrukcije formule ϕ sada slijedi $\pi \vDash \phi$.

Neka je $\pi \in \Sigma^\omega$ takav da vrijedi $\pi \vDash \phi$. Iz konstrukcije formule ϕ slijedi da postoji beskonačan niz vrhova $p = p_0, p_1, p_2, \ldots$ u Γ' tako da je $p_0 = v_0$, $(v_0, p_1) \in E'$ (jer π mora ispunjavati odgovarajuću formulu $g(D)$) i svaki prefiks od p je podniz puta u Γ' (jer π ispunjava odgovarajuću formulu h_0), te vrijedi $\pi = t(p)$. Iz tvrdnje 2. slijedi da je $\pi \in C$.

Iz konstrukcije formula h_0, $g(d)$, pri čemu je d obvezan put, slijedi da je formula ϕ pozitivna LTL formula izgrađena samo pomoću operatora G. ■

Budući da za svaku LTL formulu ϕ postoji NBA koji definira svojstvo koje izražava ϕ (vidi [22], [28]), iz lema 2.1.15. i 2.1.12. dobivamo točnu karakterizaciju formula jake sigurnosti koju ističemo sljedećim teoremom.

Teorem 2.1.16. *Formula* ϕ *izražava svojstvo jake sigurnosti ako i samo ako je* ϕ *pozitivna formula izgrađena samo pomoću operatora* G.

U dokazu leme 2.1.15. smo konstruirali formulu ϕ koja je izražavala svojstvo jake sigurnosti definirano automatom A. Pokazuje se da je generirana formula ϕ disjunkcija formula oblika $\psi_1 \wedge \psi_2$, pri čemu je ψ_1 formula bez temporalnih operatora, a ψ_2 je formula oblika h_0 iz dokaza leme. Time smo zapravo definirali jednu normalnu formu za formule jake sigurnosti.

2.1.5 Karakterizacija svojstava životnosti i pravednosti

U ovoj točki dajemo karakterizacije svojstava apsolutne životnosti i pravednosti. Kao što je već rečeno, karakterizacija životnosti u LTL-u ostaje otvoreno pitanje.

Lema 2.1.17. Formula LTL-a ϕ izražava svojstvo apsolutne životnosti ako i samo ako je ϕ ispunjiva, te vrijedi $L(\phi) = L(F\phi)$.

Dokaz Neka je ϕ formula koja izražava svojstvo apsolutne životnosti. Po definiciji ϕ je ispunjiva formula. Pokazat ćemo da vrijedi $L(\phi) = L(F\phi)$. Iz definicije istinitosti operatora F slijedi da ako za neki $\pi \in \Sigma^\omega$ vrijedi $\pi \vDash \phi$ tada vrijedi i $\pi \vDash F\phi$, tj. vrijedi $L(\phi) \subseteq L(F\phi)$. Neka je sada $\pi \in \Sigma^\omega$ takav da vrijedi $\pi \vDash F\phi$. Znači, postoji

$i \geq 0$, tako da vrijedi $\pi^i \vDash \phi$. Kako ϕ izražava svojstvo apsolutne životnosti, slijedi $\pi(i-1)\pi^i \vDash \phi$, tj. vrijedi $\pi \vDash \phi$ i $L(F\phi) \subseteq L(\phi)$.

Pretpostavimo da je ϕ ispunjiva, te da vrijedi $L(\phi) = L(F\phi)$. Neka je $\pi \in \Sigma^\omega$ takav da vrijedi $\pi \vDash \phi$. Iz definicije istinitosti operatora F slijedi, za svaki $t \in \Sigma^*$ vrijedi $t\pi \vDash F\phi$. Kako je $L(\phi) = L(F\phi)$, vrijedi i $t\pi \vDash \phi$. Znači, formula ϕ izražava svojstvo apsolutne životnosti. ■

Možemo primijetiti da formula koja izražava svojstvo *live-resource,* definirano u točki 2.1.3., zadovoljava lemu 2.1.17. Sada dajemo karakterizaciju LTL formula koje izražavaju svojstva stabilnosti, tj. svojstva koja su zatvorena na sufikse.

Lema 2.1.18. Formula LTL-a ϕ izražava svojstvo stabilnosti ako i samo ako vrijedi $L(\phi) = L(G\phi)$.

Dokaz Neka je ϕ formula stabilnosti i $\pi \in L(\phi)$. Iz definicije svojstva stabilnosti slijedi: $\pi^i \vDash \phi$, $\forall i \geq 0$, tj. vrijedi $\pi \vDash G\phi$. Znači, ako $\pi \in L(\phi)$ onda $\pi \in L(G\phi)$, tj. vrijedi $L(\phi) \subseteq L(G\phi)$. Ako je formula $G\phi$ istinita na nekom putu π, tada iz definicije operatora G slijedi da je i ϕ istinita na putu π, tj. vrijedi $L(G\phi) \subseteq L(\phi)$. Stoga, vrijedi $L(\phi) = L(G\phi)$.

Obrnuto, pretpostavimo da za LTL formulu ϕ vrijedi $L(\phi) = L(G\phi)$. Neka je put π takav da vrijedi $\pi \vDash \phi$. Iz pretpostavke slijedi, $\pi \in L(G\phi)$, tj. vrijedi $\pi \vDash G\phi$. Po definiciji istinitosti operatora G slijedi: $\pi^i \vDash \phi$, $\forall i \geq 0$. Znači, $\forall \pi \in L(\phi)$ vrijedi $\pi^i \vDash \phi$, $\forall i \geq 0$, tj. formula ϕ izražava svojstvo stabilnosti. ■

Korištenjem prethodne leme, možemo vidjeti da formule Gp, $G(p \vee Gq)$ *i* GFp izražavaju svojstva stabilnosti. Sada dajemo karakterizaciju formula koje izražavaju svojstva pravednosti. Prisjetimo se, formula pravednosti je LTL formula koja izražava ujedno i svojstvo apsolutne životnosti, i svojstvo stabilnosti. Sada direktno iz lema 2.1.18. i 2.1.19. slijedi tvrdnja teorema 2.1.20.

Teorem 2.1.20. Formula LTL-a ϕ izražava svojstvo pravednosti ako i samo ako je ϕ ispunjiva, te vrijedi $L(\phi) = L(F\phi) = L(G\phi)$.

Možemo primijetiti da formula koja izražava svojstvo *channel-fairness* zadovoljava teorem 2.1.20. Svojstvo *channel-fairness* se još obično naziva i svojstvo jake pravednosti. Slabija verzija svojstva *channel-fairness,* koja se obično naziva *weak channel-fairness*, može se izraziti sljedećom formulom pravednosti:

$$FGsend \rightarrow GFreceive.$$

Još jedno važno svojstvo pravednosti je tzv. *process-fairness.* Ako propozicionalna varijabla ex predstavlja izvršenje jednog koraka *procesa 1,* tada formula $GFex$ utvrđuje da će se *proces 1* izvršiti beskonačno mnogo puta.

2.1.6 Dekompozicija LTL-a

U ovoj točki ćemo pokazati da se svaka LTL formula može prikazati kao konjunkcija formule sigurnosti i formule životnosti (eng. safety-liveness decomposition). Prilikom dokazivanja rezultata iz ove točke koristit ćemo Büchijeve automate koje smo definirali u točki 2.1.4.

Skup svojstava nazivamo **familija.** Familiju svih svojstava označavamo s Π, familiju svih svojstava koje možemo izraziti LTL formulama označavamo s $\mathcal{L}$. Ako je $\mathcal{F}$ neka familija, tada s $\mathcal{F}^s$ označavamo sva svojstva sigurnosti koja pripadaju familiji $\mathcal{F}$, tj. $\mathcal{F}^s = \{C \in \mathcal{F} \mid C \; je \; svojstvo \; sigurnosti\}$. Analogno, s $\mathcal{F}^l$ označavamo sva svojstva životnosti koja pripadaju familiji $\mathcal{F}$. Primijetimo, Π^s odgovara familiji svih svojstava sigurnosti, a Π^l odgovara familiji svih svojstava životnosti. Familija $\mathcal{F}$ je zatvorena za n-mjesnu funkciju $f\colon \Pi^n \longrightarrow \Pi$, $n \geq 1$, ako za sve $C_1, C_2, \dots, C_n \in \mathcal{F}$ vrijedi $f(C_1, C_2, \dots, C_n) \in \mathcal{F}$.

Od posebne zanimljivosti će nam biti unarna funkcija **topološkog zatvorenja** $\lceil * \rceil : \Pi \longrightarrow \Pi$, definirana sa $\lceil C \rceil = \bigcap\{S \in \Pi^s \mid C \subseteq S\}$. Drugim riječima, $\lceil C \rceil$ je najmanje svojstvo sigurnosti koje sadrži svojstvo C. Također, definiramo funkciju **komplementa** $\bar{*} : \Pi \longrightarrow \Pi$, $\bar{C} = \Sigma^\omega \backslash C$. Ako je familija zatvorena na $\cap$ i $\bar{*}$, onda je zatvorena i na $\cup$. Sada definiramo što znači da neka familija prihvaća dekompoziciju na sigurnost i životnost.

Definicija 2.1.21. *Kažemo da familija $\mathcal{F}$ prihvaća dekompoziciju na životnost i sigurnost ako vrijedi*

$$(\forall C \in \mathcal{F})(\exists S \in \mathcal{F}^s)(\exists L \in \mathcal{F}^l)(C = S \cap L).$$

Znači, familija prihvaća dekompoziciju na životnost i sigurnost ako se svako svojstvo iz familije može prikazati kao presjek svojstva sigurnosti i svojstva životnosti. Svojstva sigurnosti i životnosti se također mogu definirati metrički. Udaljenost između dva puta možemo mjeriti na osnovu najmanjeg prefiksa u kojem se ta dva puta razlikuju. Formalno, definiramo **metriku** $\boldsymbol{d}$ na prostoru svih puteva Σ^ω tako da vrijedi $d(\pi, \pi) = 0$ i $d(\pi, \sigma) = 2^{-i}$, gdje je $i = \min\{j \in \mathbb{N} \mid \pi(j) \neq \sigma(j)\}$. Udaljenost između puta π i svojstva C je po definiciji jednaka $d(\pi, C) = \inf\{d(\pi, \sigma) \mid \sigma \in C\}$. Intuitivno, promatrajući definiciju svojstva sigurnosti i životnosti možemo reći da je skup puteva S svojstvo sigurnosti ako S sadržava sve puteve koji su mu proizvoljno blizu, a

skup puteva L je svojstvo životnosti ako mu je svaki put proizvoljno blizu. Ova zapažanja možemo sumirati sljedećom propozicijom.

Propozicija 2.1.22. *Skup* $S \subseteq \Sigma^\omega$ *je svojstvo sigurnosti ako i samo ako vrijedi* $\{\pi \in \Sigma^\omega \mid d(\pi, S) = 0\} \subseteq S$. *Skup* $L \subseteq \Sigma^\omega$ *je svojstvo životnosti ako i samo ako* $\forall \pi \in \Sigma^\omega$ *vrijedi* $d(\pi, L) = 0$.

Ovakve karakterizacije svojstava sigurnosti i životnosti odgovaraju karakterizaciji zatvorenih i gustih skupova u metričkim prostorima. Stoga, u topologiji induciranoj metrikom d, svojstva sigurnosti odgovaraju zatvorenim skupovima, a svojstva životnosti gustim skupovima. Slijedi da su svojstva sigurnosti zatvorena na (beskonačne) presjeke, a svojstva životnosti na unije. Koristeći ove topološke karakterizacije sigurnosti i životnosti može se pokazati da vrijedi sljedeći teorem (vidi [1]).

Teorem 2.1.23. *Za svako svojstvo* $C \in \Pi$, *postoji svojstvo* $S \in \Pi^s$ *i svojstvo* $L \in \Pi^l$ *tako da vrijedi* $C = S \cap L$.

Znači, familija svih svojstava prihvaća dekompoziciju na sigurnost i životnost, tj. svako svojstvo se može prikazati kao presjek svojstva sigurnosti i svojstva životnosti. Specijalno, svako svojstvo $C \in \mathcal{L}$ se može prikazati kao presjek svojstva sigurnosti i svojstva životnosti. No, nas zanima da li se svaki $C \in \mathcal{L}$ može prikazati kao presjek svojstva sigurnosti S i svojstva životnosti L, tako da vrijedi $S, L \in \mathcal{L}$. Drugim riječima, zanima nas da li skup svih svojstava koja se mogu izraziti u LTL-u prihvaća dekompoziciju na sigurnost i životnost. Iz sljedeće propozicije vidimo zašto smo rekli da nam je funkcija topološkog zatvorenja od posebne zanimljivosti.

Propozicija 2.1.24. *Neka je* $\mathcal{F}$ *familija zatvorena na* $\cap$, $\overline{*}$ *i* $\lceil * \rceil$, *tada* $\mathcal{F}$ *prihvaća dekompoziciju na sigurnost i životnost. Točnije, za svaki* $C \in \mathcal{F}$ *vrijedi* $C = \lceil C \rceil \cap (\overline{\lceil C \rceil} \cup C)$, *gdje je* $\lceil C \rceil \in \mathcal{F}^s$, *a* $\overline{\lceil C \rceil} \cup C \in \mathcal{F}^l$.

Dokaz propozicije se može vidjeti primjerice u [1]. Kako je $\mathcal{L}$ zatvoren na $\cap$ i $\overline{*}$ (jer je LTL je zatvoren na konjunkciju i negaciju), ukoliko se pokaže da je zatvoren i za $\lceil * \rceil$, onda prema prethodnoj propoziciji slijedi da $\mathcal{L}$ prihvaća dekompoziciju na sigurnost i životnost. Uspostavljanjem veze između Büchijevih automata i LTL-a pokazat ćemo da je $\mathcal{L}$ zatvoren za $\lceil * \rceil$. Drugim riječima, pokazat ćemo da za svaku LTL formulu ϕ postoji formula, u oznaci $\lceil \phi \rceil$, tako da vrijedi $\lceil L(\phi) \rceil = L(\lceil \phi \rceil)$.

Za $t \in \Sigma^+$ i funkciju prijelaza δ nekog automata definiranog na Σ, pišemo $(s, t, s') \in \delta^+$ ako postoji niz stanja $s_0, s_1, \dots, s_{|t|}$ tako da je $s = s_0$, $s' = s_{|t|}$ i $\delta(s_i, t_i) = s_{i+1}$, $\forall i, 0 \leq i < |t|$. Za stanja $p, q \in Q$, definiramo skup $L_{p,q} = \{t \in \Sigma^+ \mid (p, t, q) \in \delta^+\}$ svih nepraznih tragova čijim čitanjem promatrani automat može doći iz stanja p u stanje q. Kažemo da je stanje q **dostižno** iz stanja p ako je $L_{p,q} \neq \emptyset$.

Prisjetimo se, za put $\pi \in \Sigma^{\omega}$ s $\pi^i, i \geq 0$, smo označavali i-ti sufiks puta π. Za trag $t \in \Sigma^+$, s $t^m, m > 0$, ćemo označavati konkatenciju traga t sa samim sobom m puta. Kažemo da je automat A **aperiodičan** ako vrijedi: $(\forall q \in Q)(\forall t \in \Sigma^+)(\forall m \geq 1)\left(\left(t^m \in L_{q,q}\right) \Longrightarrow \left(t \in L_{q,q}\right)\right)$. Familiju svih svojstava koje prepoznaju Büchijevi automati označavamo s Ω. Kao što smo već napomenuli u prethodnoj točki, za svaku LTL formulu ϕ postoji NBA koji definira svojstvo koje izražava ϕ. No, pokazuje se da pomoću NBA možemo definirati svojstva koja ne možemo izraziti u LTL-u, tj. vrijedi $\mathcal{L} \subset \Omega$ (vidi [28]).

Familija $\mathcal{L}$ se može okarakterizirati pomoću tzv. ***counter-free*** automata (automati bez brojača). Pokazuje se da je Büchijev automat *counter-free* ako i samo ako je aperiodičan (vidi [5]). Sljedećim teoremom uspostavljamo vezu između LTL-a i *counter-free* automata.

Teorem 2.1.25. *Neka je $C \subseteq \Sigma^{\omega}$. Postoji LTL formula ϕ tako da je $L(\phi) = C$ ako i samo ako postoji counter-free Büchijev automat A tako da je $L(A) = C$.*

Dokaz teorema je dan u [5]. Sada dajemo definiciju reduciranih automata.

Definicija 2.1.26. *Kažemo da je automat A **reduciran** ako je iz svakog stanja automata A dostižno neko prihvatljivo stanje automata A.*

Svaki Büchijev automat A se može reducirati tako da dobiveni automat prihvaća isti jezik. Ukoliko iz početnog stanja automata A nije dostižno neko stanje prihvaćanja, tj. ako vrijedi $L(A) = \emptyset$, onda je automat koji ne sadrži nijedno stanje odgovarajući reducirani automat za A. Inače, iterativno odstranjujemo iz skupa stanja Q svako stanje $s \in F$ iz kojeg nije dostižno barem jedno stanje prihvaćanja i sukladno tome restringiramo funkciju prijelaza δ. Zatim, iz Q izbacujemo sva stanja $s \in Q \backslash F$ iz kojih nije dostižno barem jedno stanje prihvaćanja i sukladno tome restringiramo δ. Dobiveni automat je reduciran i prihvaća isti jezik kao i A.

Za Büchijev automat A, automat koji prihvaća $\lceil L(A) \rceil$ se može konstruirati reduciranjem automata A i proširivanjem skupa stanja prihvaćanja na sva stanja automata. Dokaz sljedećeg teorema je dan u [2].

Teorem 2.1.27. *Neka je A Büchijev automat i neka je $A_R = (\Sigma, Q, \delta, q_0, F)$ automat koji dobijemo reduciranjem automata A. Topološko zatvorenje jezika (svojstva) kojeg definira automat A prihvaća automat $(\Sigma, Q, \delta, q_0, Q)$.*

Budući je Ω zatvoren na $\cap$, $\overline{*}$ (vidi [9]), direktno iz propozicije 2.1.24. i teorema 2.1.27. slijedi tvrdnja korolara 2.1.28.

Korolar 2.1.28. Za svaki $C \in \Omega$ postoji $C_s \in \Omega^s$ i $C_l \in \Omega^l$ tako da vrijedi $C = C_s \cap C_l$.

Znači, familije Π i Ω prihvaćaju dekompoziciju na sigurnost i životnost. Sada želimo dokazati da $\mathcal{L}$ prihvaća dekompoziciju na sigurnost i životnost. Primijetimo da ukoliko dokažemo da reduciranjem *counter-free* automata opet dobivamo *counter-free* automate, tada možemo opet iskoristiti teorem 2.1.27. i propoziciju 2.1.24. da bi dokazali da $\mathcal{L}$ prihvaća dekompoziciju na sigurnost i životnost.

Lema 2.1.29. *Neka je A counter-free Büchijev automat i neka je A' Büchijev automat koji dobijemo reduciranjem automata A. Tada je A' counter-free.*

Dokaz Ako q_0 (početno stanje) nije stanje automata A', tj. ako iz početnog stanja nije dostižno stanje prihvaćanja, onda $L(A) = \emptyset$. Budući je po definiciji automat reduciran ako je iz svakog njegovog stanja dostižno neko stanje prihvaćanja, tada vidimo da automat A' ne sadrži ni jedno stanje, pa je stoga *counter-free*. Inače, pretpostavimo da automat $A' = (\Sigma, Q', \delta', q_0, F')$ nije *counter-free.* Tada postoji trag $t \in \Sigma^+$, $m > 1$, $p \in Q'$ tako da vrijedi $t^m \in L'_{p,p}$ i $t \notin L'_{p,p}$ u A'. Kako je A *counter-free*, vrijedi $t \in L_{p,p}$ u A. Stoga, postoji niz stanja $s_0, s_1, \dots, s_{|t|}$, pri čemu je $s_0 = s_{|t|} = p$ i $(s_i, t_i, s_{i+1}) \in \delta, \forall i, 0 \leq i < |t|$. Budući da $t \notin L'_{p,p}$, mora postojati stanje s_i koje nije stanje automata A'. To znači da u A ne postoji stanje prihvaćanja koje je dostižno iz stanja s_i. Budući je stanje p dostižno iz s_i, slijedi da ni iz p nije dostižno neko stanje prihvaćanja. No, ako iz p nije dostižno neko stanje prihvaćanja u A, onda iz p nije dostižno neko stanje prihvaćanja u A'. Znači, dobili smo kontradikciju budući da je A' reduciran. ■

Sada smo spremni dokazati da je LTL zatvoren na operaciju $\lceil * \rceil$. Zatim ćemo kao korolar dobiti da LTL (točnije $\mathcal{L}$) prihvaća dekompoziciju na sigurnost i životnost.

Teorem 2.1.30. *Neka je ϕ proizvoljna LTL formula. Tada postoji LTL formula, koju označavamo sa $\lceil \phi \rceil$, tako da je $L(\lceil \phi \rceil) = \lceil L(\phi) \rceil$.*

Dokaz Iz teorema 2.1.25. slijedi da postoji *counter-free* automat A tako da je $L(A) = L(\phi)$. Neka je A_s automat koji dobijemo reduciranjem automata A, i *označavanjem* svih njegovih stanja prihvaćajućim. Iz teorema 2.1.27., slijedi da vrijedi $L(A_s) = \lceil L(A) \rceil = \lceil L(\phi) \rceil$. Iz definicije aperiodičnosti vidimo da mijenjanjem skupa stanja prihvaćanja Büchijevog automata ne utječemo na njegovu aperiodičnost. Stoga, iz leme 2.1.29. slijedi da je Büchijev automat A_s *counter-free*. Iz teorema 2.1.25. slijedi tvrdnja teorema. ■

Budući da je LTL zatvoren na konjunkciju i negaciju, $\mathcal{L}$ je zatvoren na presjeke i komplemet. Direktno iz propozicije 2.1.24. i teorema 2.1.25. slijedi tvrdnja korolara 2.1.31.

Korolar 2.1.31. Za svaki $C \in \mathcal{L}$ postoji $C_s \in \mathcal{L}^s$ i $C_l \in \mathcal{L}^l$ tako da vrijedi $C = C_s \cap C_l$.

Znači, $\mathcal{L}$ prihvaća dekompoziciju na sigurnost i životnost. Slijedi da za svaku LTL formulu ϕ postoji formula sigurnosti ϕ_s i formula životnosti ϕ_l tako da vrijedi $\phi \equiv \phi_s \wedge \phi_l$. Iako je poznato da za svaku LTL formulu postoji *counter-free* automat koji prepoznaje jezik dane formule (teorem 2.1.25.), ipak nije poznat neki algoritam za konstrukciju tog automata. Poznati su algoritmi kojima se za danu LTL formulu može konstruirati Büchijev automat koji prepoznaje jezik te formule (vidi npr. [8]). Korištenjem ovih algoritama te teorema 2.1.27., za danu LTL formulu ϕ se može konstruirati *counter-free* automat koji prepoznaje $L(\lceil\phi\rceil)$, a onda i formula $\lceil\phi\rceil$ koja definira $L(\lceil\phi\rceil)$ (vidi [19]). Znači, ako je ϕ formula sigurnosti, možemo konstruirati *counter-free* automat koji prepoznaje $L(\phi)$.

Napomenimo još da postoje algoritmi za prepoznavanje formula sigurnosti, odnosno životnosti. Sistla (vidi [22]) daje jedan PSPACE složen algoritam koji za danu formulu odlučuje da li ona izražava svojstvo sigurnosti. Također, opisan je i jedan algoritam složenosti 2EXP (dvostruko eksponencijalan – DTIME(2^{2^n})) za prepoznavanje formula životnosti.

2.2 Provjera modela (eng. model checking)

U ovoj točki pokazujemo konkretno na primjerima kako se LTL upotrebljava za verifikaciju sustava. Postoje razni načini formalne verifikacije, koji se razlikuju s obzirom na:

- *okvir (eng. framework) za modeliranje sustava* (obično se radi o nekakvom jeziku za opisivanje)
- *specifikacijski jezik* (to je jezik za opisivanje svojstava koje treba verificirati; mi ćemo koristiti jezik LTL-a)
- *verifikacijsku metodu* (to je metoda pomoću koje utvrđujemo da li opis sustava zadovoljava specifikaciju).

Verifikacijske metode se mogu klasificirati prema sljedećim kriterijima:

- **Pristup verifikaciji.**
  Verifikacija može biti zasnovana na modelima ili na formalnim dokazima. U tehnikama zasnovanim na formalnim dokazima, sustav se opisuje skupom formula Γ (u odgovarajućoj logici) a specifikacija svojstava je također zadana u obliku formule ϕ. Verifikacija se provodi izvođenjem formule ϕ iz skupa pretpostavki Γ, tj. dokazujemo da vrijedi $\Gamma \vdash \phi$. Ovakvo dokazivanje obično zahtjeva puno truda i znanja od strane izvršitelja.
  U tehnikama zasnovanima na modelima, sustav je reprezentiran modelom $\mathcal{M}$ u odgovarajućoj logici. Svojstva su ponovo zadana u obliku formule ϕ, a verifikaciju se provodi provjeravanjem istinitosti formule ϕ u modelu $\mathcal{M}$, tj. provjerava se da li vrijedi $\mathcal{M} \vDash \phi$. Ovakvo provjeravanje se obično obavlja automatski za konačne modele.

- **Stupanj automatiziranosti.**
  U ekstremnim slučajevima, metode za verifikaciju mogu biti potpuno automatizirane, odnosno potpuno manualne. Većina metoda je poluautomatizirana i zahtijeva aktivno sudjelovanje stručnih osoba prilikom izvršavanja.

- **Razina verificiranosti.**
  Specifikacija može opisivati pojedina svojstva sustava, ili može opisivati ponašanje sustava u potpunosti. Verificiranje ponašanja sustava u potpunosti je obično prezahtijevno.

- **Domena aplikacije sustava.**
  Sustav koji se verificira može biti hardware ili softver; sekvencijalni ili paralelni; reaktivni ili izvršni; ... Reaktivni sustav je onaj koji se vrti u beskonačnoj petlji i reagira na upite iz okoline (npr. operacijski sustavi), dok izvršni sustav obavlja neki zadatak i zatim prestaje s radom.

- **Mjesto u razvojnom procesu.**
  Verifikacija se može odvijati tokom ili nakon razvoja sustava. Greške otkrivene tokom razvoja sustava se lakše ispravljaju.

Metoda koju ćemo proučavati u ovoj točki zasniva se na modelima i zove se *provjera modela* (eng. model checking). Ovu metodu koristimo za provjeru pojedinih svojstva paralelnih sustava, obično na kraju razvojnog procesa. Provjera modela se često koristi u praksi, jer je u velikoj mjeri automatizirana i jednostavna za provedbu. No, konkretno rješenje kojim se automatski provjerava da li specificirana formula zadovoljava model je teško efikasno implementirati.

Točnije, u ovoj točki ćemo kroz primjere pokazati kako se LTL može koristiti u provjeri modela (eng. LTL model checking). Modeli koji se provjeravaju će odgovarati tranzicijskim sustavima, a svojstva koja se provjeravaju izražavat ćemo pomoću LTL formula. Verifikacija sustava LTL provjerom modela se obavlja tako da:

- modeliramo sustav koristeći jezik alata za provjeru modela (eng. *model checker*), čime dobivamo model $\mathcal{M}$,
- izrazimo svojstvo koje provjeravamo u obliku LTL formule ϕ,
- pokrenemo alat za provjeru modela s ulazima $\mathcal{M}$ i ϕ.

Alat za provjeru modela daje potvrdan odgovor ako je formula ϕ istinita u modelu $\mathcal{M}$, tj. ako vrijedi $\mathcal{M} \vDash \phi$. Inače, alat vraća negativan odgovor. U slučaju negativnog odgovora, većina alata za provjeru modela vraća i trag t u modelu $\mathcal{M}$ za koji ne vrijedi formula ϕ $(\mathcal{M}, t \nvDash \phi)$. Ovo svojstvo vraćanja tragova koji uzrokuju greške je vrlo korisno prilikom pronalaženja uzroka greški te njihovog ispravljanja.

Spomenimo još da je jedna od zastupljenijih alternativnih metoda baziranih na LTL-u tzv. deduktivna verifikacija. Kod ispitivanja korektnosti rada sustava deduktivnom verifikacijom svojstva sustava se izražavaju u obliku LTL formula koje dokazujemo formalno u jednom deduktivnom sustavu LTL-a. No, ovakav pristup verificiranju se pokazao kao dosta zahtijevan i složen za primjenu u praksi. Kod malo većih, složenijih programa postaje jako teško provoditi formalne dokaze korektnosti.

2.2.1 Primjeri

Primjer 2.2.1. Prvi primjer, koji ćemo ujedno i najdetaljnije obraditi, je problem *međusobnog isključivanja* (eng. *mutual exclusion*) za paralelne procese. Kada paralelni procesi dijele neki resurs (npr. datoteku na disku), možda je potrebno osigurati da ti procesi ne mogu pristupati tom dijeljenom resursu u isto vrijeme. Npr. u bankovnim sustavima prilično je bitno da stanju bankovnog računa ne može istovremeno pristupati više procesa jer bi primjerice moglo doći do neispravnog obračunavanja prilikom istovremene uplate i isplate.

U svrhu rješavanja ovog problema identificiramo određene dijelove koda svakog procesa kao *kritične odsječke* (eng. *critical section*). U svakom trenutku najviše jedan proces može biti u svojem kritičnom odsječku. Kritični odsječci bi trebali odgovarati dijelovima koda u kojima proces pristupa dijeljenom resursu i pritom bi taj dio trebao biti što manji, tako da ne dolazi do bespotrebnog zadržavanja ostalih procesa koji možda čekaju na pristup dijeljenom resursu. Znači, za rješavanje ovog problema potrebno je pronaći *protokol* za određivanje procesa koji može ući u svoj kritični odsječak u danom trenutku. Kada pronađemo rješenje za koje mislimo da je ispravno, verificiramo ga tako

da provjerimo da li zadovoljava neka željena svojstva. Svojstva koja mi želimo da naš protokol zadovoljava su:

1) **Sigurnost**: U svakom trenutku najviše jedan proces može biti u svojem kritičnom odsječku.

Ovo svojstvo nije dovoljno, budući da protokol koji trajno spriječava svaki proces da uđe u svoju kritičnu sekciju zadovoljava svojstvo sigurnosti, ali očito nije baš i koristan. Stoga, zahtijevamo da naše rješenje zadovoljava i sljedeće uvjete koje sada navodimo.

2) **Životnost**: Ako neki proces zahtijeva pristup kritičnom odsječku, taj zahtjev će eventualno biti i odobren.

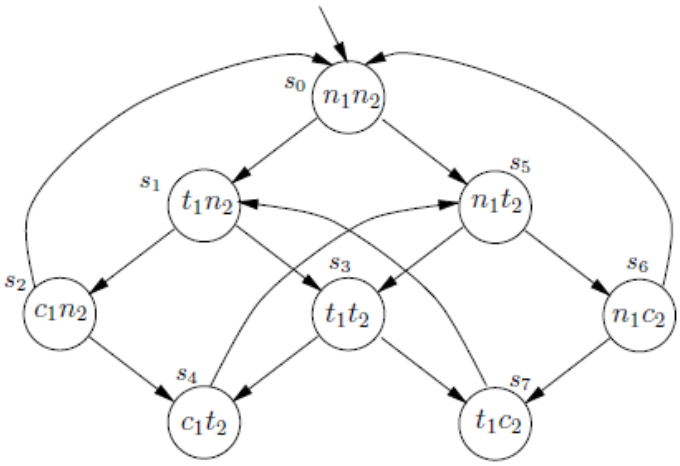
3) **Dostupnost**: Proces uvijek može zahtijevati pristup svojem kritičnom odsječku.

Neki protokoli uspostavljaju određeni poredak procesa prema kojem dozvoljavaju pristup kritičnom odsječku. U takvim slučajevima i-ti proces ne može dobiti pristup kritičnom odsječku sve dok proces $i-1$ ne zatraži pristup kritičnom odsječku te izađe iz njega. Ovakvo rješenje je nepoželjno, stoga zahtijevamo da naš protokol zadovoljava još jedan uvjet:

4) **Bez fiksnog redoslijeda**: Procesi ne moraju dobivati pristup kritičnom odsječku u određenom redoslijedu.

Promotrimo prvo jedan protokol zadan tranzicijskim sustavom $\mathcal{M}$ sa slike 2.1 (na sljedećoj stranici). Tranzicijski sustav $\mathcal{M}$ modelira dva procesa čije se operacije međusobno isprepleću. U danom trenutku samo jedan od procesa se može izvršavati. S obzirom na zahtjeve problema, procese modeliramo tako da razlikujemo samo sljedeća ponašanja procesa:

- proces zahtijeva pristup kritičnom odsječku (t)
- proces je u kritičnom odsječku (c)
- proces nije u kritičnom odsječku, niti zahtijeva pristup kritičnom odsječku (n)



SLIKA 2.1: PRVI POKUŠAJ MODELIRANJA PROTOKOLA ZA MEĐUSOBNO ISKLJUČIVANJE

Na početku rada nijedan proces nije u kritičnom odsječku, niti traži pristup kritičnom odsječku (stanje s_0). Sada želimo izraziti prethodno navedena svojstva kao LTL formule. Pri čemu nas naravno zanima zadovoljava li sustav svojstva od početka rada, tj. zanima nas vrijede li odgovarajuće LTL formule u inicijalnom stanju s_0. Redom komentiramo svojstva.

Sigurnost: Vidimo da ovo svojstvo odgovara skupu svih puteva u modelu $\mathcal{M}$ za koje se ni u jednom trenutku ne pojavljuje stanje označeno varijablom c_1, a ujedno i varijablom c_2. Očito, LTL formula $G\neg(c_1 \wedge c_2)$ izražava ovo svojstvo. Iz slike 2.1 vidimo da model $\mathcal{M}$ zadovoljava ovo svojstvo u početnom stanju s_0, tj. vrijedi $\mathcal{M}, s_0 \vDash G\neg(c_1 \wedge c_2)$.

Životnost: Možemo vidjeti da formula $G(t_1 \rightarrow Fc_1) \wedge G(t_2 \rightarrow Fc_2)$ izražava ovo svojstvo. No, na putu $s_0 \mapsto s_1 \mapsto s_3 \mapsto s_7 \mapsto s_1 \mapsto s_3 \mapsto s_7 \mapsto \cdots$, promatrana formula nije istinita jer nijedno stanje nije označeno varijablom c_1, tj. $\mathcal{M}, s_0 \nvDash G(t_1 \rightarrow Fc_1) \wedge G(t_2 \rightarrow Fc_2)$.

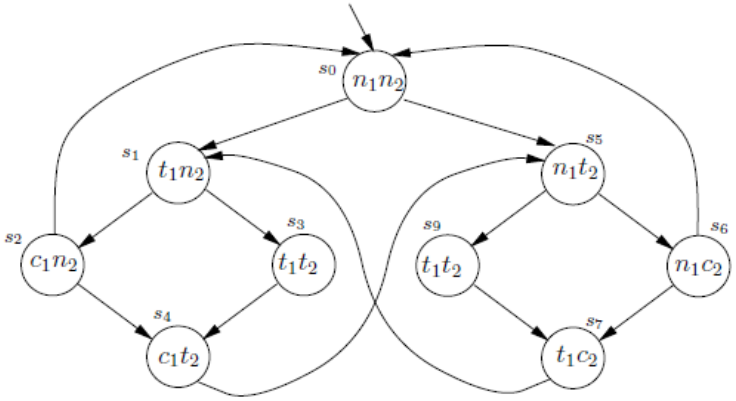
Dostupnost: Uzmimo u obzir samo proces 1 (proces opisan varijablama n_1, c_1, t_1). Nas zapravo zanima postoji li za svaki trag t u $\mathcal{M}$ koji započinje u stanju s_0 i koji završava u stanju koje je označeno varijablom n_1, put $\pi \in \mathcal{M}$ na kojem se pojavljuje stanje označeno varijablom c_1 i vrijedi $t\pi \in \mathcal{M}$. Ovaj primjer nam zapravo pokazuje jednu ograničenost LTL -a, jer ovakvo egzistencijalno kvantificiranje na skupovima puteva se ne može izraziti u LTL -u. Dakle, dostupnost ne možemo izraziti LTL formulama.

Bez fiksnog redoslijeda: Primijetimo da je ovaj zahtjev ispunjen ako i samo ako postoji put na kojem između dva ulaska u kritični odsječak jednog od procesa, ne

postoji stanje u kojem drugi proces ulazi u svoj kritični odsječak. Vidimo da sada opet imamo egzistencijalno kvantificiranje gdje pokušavamo utvrditi postojanje nekog puta. No, u ovom slučaju možemo izraziti komplementarno svojstvo. Npr. za proces 1 to znači da za svaki put vrijedi da između svaka dva ulaska jednog procesa u svoj kritični odsječak, postoji stanje u kojem drugi proces ulazi u svoj kritični odsječak. Za proces 1 ovo svojstvo možemo izraziti sljedećom LTL formulom: $G(c_1 \rightarrow c_1\, W\, (\neg c_1 \wedge \neg c_1\, W\, c_2))$. Budući da ova formula izražava komplementarno svojstvo, ako vrijedi $\mathcal{M}, s_0 \not\models G(c_1 \rightarrow c_1\, W\, (\neg c_1 \wedge \neg c_1\, W\, c_2))$, onda smo dokazali da postoji put iz s_0 na kojem između dva ulaska u kritični odsječak jednog od procesa, ne postoji stanje u kojem drugi proces ulazi u svoj kritični odsječak. Na slici 1 možemo primijetiti da put $\pi = s_0 \mapsto s_5 \mapsto s_3 \mapsto s_4 \mapsto s_5 \mapsto s_3 \mapsto s_4 \mapsto \cdots$ ima traženo svojstvo. Stoga je orginalni zahtjev da nema fiksnog redoslijeda ispunjen.

Primijetimo da smo u četvrtom uvjetu mogli izraziti zahtjev za postojanjem nekog puta svođenjem na komplement koji se onda odnosi na sve puteve. No, u trećem uvjetu nismo mogli izraziti zahtjev za postojanjem određenog puta. Razlog tome je što je treći uvjet ipak složeniji. Prvo imamo univerzalnu kvantifikaciju a zatim egzistencijalnu, tj. za svaki trag nas zanima postoji li određeni put. U ovom slučaju, promatranjem komplementa ne dobivamo ništa jer se tada univerzalni kvantifikator pretvara u egzistencijalni.

Kao što smo vidjeli, ovaj model ne predstavlja zadovoljavajuće rješenje problema međusobnog isključivanja jer uvjet životnosti nije zadovoljen. Razlog tome je nedeterminizam protokola prilikom određivanja procesa koji treba dobiti pravo pristupa kritičnom odsječku kada oba procesa zahtijevaju pristup. Zbog ovoga je moguće da protokol kontinuirano favorizira jedan proces nad drugim. Problem je u tome što stanje s_3 ne raspoznaje koji je proces prvi tražio pristup kritičnom odsječku. Ovaj problem se može riješiti tako da stanje s_3 podijelimo u dva nova stanja, kao što je prikazano na slici 2.2.



SLIKA 2.2: DRUGI POKUŠAJ MODELIRANJA PROTOKOLA ZA MEĐUSOBNO ISKLJUČIVANJE

U ovom novom modelu vidimo da stanja s_3 i s_9 odgovaraju stanju s_3 iz starog modela. No, kada oba procesa zahtijevaju pristup kritičnom odsječku sada znamo koji je od njih prvi tražio pristup. Primijetimo da su stanja s_3 i s_9 oba označena istim propozicionalnim varijablama: t_1 i t_2. Prema definiciji tranzicijskih sustava dozvoljeno je imati stanja koja su označena istim varijablama. No, malo je neobično da dva zapravo jednaka stanja imaju različita ograničenja s obzirom na prijelaze u druga stanja. Ovakve *probleme* možemo riješiti dodavanjem novih (skrivenih) propozicionalnih varijabli koje nam služe isključivo za razlikovanje stanja.

Možemo provjeriti da novi model sada zadovoljava sva željena svojstva. (Treće svojstvo, koje nismo mogli izraziti u LTL-u, možemo provjeriti samo neformalno.) Također, napomenimo da je ovaj model možda i previše pojednostavljen jer se pretpostavlja da će u svakom koraku jedan od procesa prijeći u novo stanje. No, ukoliko uključimo mogućnost da procesi mogu ostati u istom stanju, tj. u graf modela uključimo strelice koje izlaze iz istog stanja na koje i pokazuju, tada model neće zadovoljavati željeno svojstvo životnosti. U realnim situacijama ovaj problem se rješava tako da se prilikom zadavanja modela u jeziku alata za provjeru modela postave tzv. uvjeti pravednosti koji osiguravaju da će sustav eventualno prijeći u novo stanje.

Primjer 2.2. Pogledajmo sada primjer jedne popularne logičke zagonetke u kojoj je cilj s jedne strane obale čamcem prevesti na drugu stranu obale vuka, kozu i zelje. Znači, zanima nas da li je moguće čamcem prevesti vuka, kozu i zelje na drugu stranu obale, pri čemu postoje određena ograničenja, odnosno uvjeti:

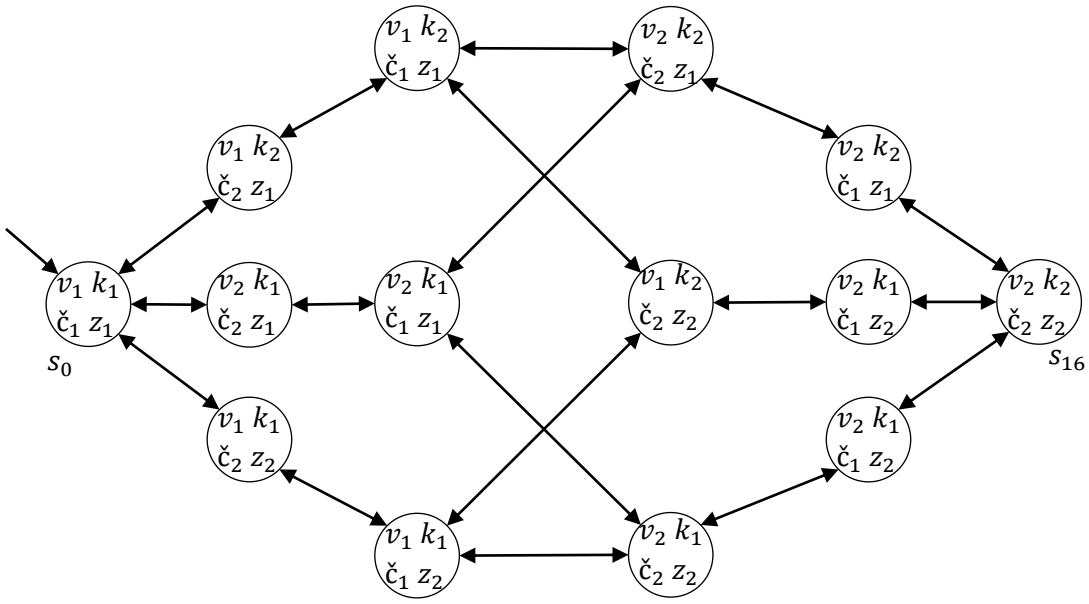
(1) čamcem možemo prevoziti samo po jednog putnika,
(2) na istoj strani obale nikad ne smiju ostati sami vuk i koza,
(3) na istoj strani obale nikad ne smiju ostati sami koza i zelje.

Ovo je zapravo jedan *problem planiranja*, ali se može riješiti provjerom modela. Tranzicijski sustav će se sastojati od stanja koja će biti označena propozicionalnim varijablama v_i, k_i, z_i i $\check{c}_i$, $i \in \{1,2\}$. Pri čemu smatramo da ako je neko stanje označeno varijablom v_1, tada je vuk na inicijalnoj (krivoj) strani obale. Ako je stanje označeno varijablom v_2, tada smatramo da je vuk na pravoj strani obale. Analogno vrijedi za (k)ozu, (z)elje i (č)amac. Početno stanje modela će biti onda označeno varijablama $v_1, \check{c}_1, k_1$ i z_1. Sada je rješavanje problema ekvivalentno pronalaženju puta u modelu sa slike 2.3 (na sljedećoj stranici) koji ispunjava formulu $\phi\ U\ \psi$, pri čemu je:

$$\phi \equiv \neg((v_1 \wedge k_1 \wedge \neg\check{c}_1 \wedge \neg z_1) \vee (v_2 \wedge k_2 \wedge \neg\check{c}_2 \wedge \neg z_2) \vee (\neg v_1 \wedge k_1 \wedge \neg\check{c}_1 \wedge z_1) \vee (\neg v_2 \wedge k_2 \wedge \neg\check{c}_2 \wedge z_2)),$$

$$\psi \equiv v_2 \wedge k_2 \wedge z_2.$$

Postojanje puta koji zadovoljava formulu $\phi\, U\, \psi$ utvrđujemo ispitivanjem vrijedi li $\mathcal{M}, s_0 \vDash \neg(\phi\, U\, \psi)$. Ukoliko postoji traženi put, tada formula $\neg(\phi\, U\, \psi)$ nije istinita u stanju s_0 modela $\mathcal{M}$.



SLIKA 2.3: MODEL PROBLEMA VUK, KOZA I ZELJE

Primjer 2.3. U ovom primjeru proučavamo jedan sustav kontrole rada semafora. Točnije, promatramo jedan pojednostavljeni sustav koji kontrolira tok prometa na raskrižju jedne glavne ceste i jedne sporedne ceste. Promatrani sustav će se sastojati od četiri varijable koje mogu poprimati dvije različite vrijednosti. Varijable sustava su semafor na glavnoj cesti, semafor na sporednoj cesti, sat i senzor za prisutnost vozila na sporednoj cesti. Semafori mogu biti crveni ili zeleni, sat može imati vrijednost *short*, *medium* ili *long*, a senzor poprima vrijednost 0 ili 1. Radi jednostavnosti, smatramo da semafor implicitno poprima žuto svjetlo prilikom promjene boje.

Senzor ima vrijednost 0 ako na sporednoj cesti nema vozila, inače ima vrijednost 1. Kad semafor na glavnoj cesti promjeni boju, sat se resetira na inicijalnu vrijednost i počinje odbrojavati. Inicijalna vrijednost sata je *short***,** nakon određenog perioda poprima vrijednost *medium*, a zatim i *long***.**

Tranzicijski sustav kojim modeliramo kontrolu rada semafora će se sastojati od stanja koja će biti označena propozicionalnim varijablama iz skupa $\{c_1, c_2, z_1, z_2, 0,1, s, m, l\}$. Intuitivno, tumačimo da ako je stanje označeno varijablom c_1 onda je na semaforu glavne ceste upaljeno crveno svjetlo, a ako je stanje označeno npr. s varijablama s $c_2, z_1, 1, l$, to onda tumačimo kao da je na semaforu sporedne ceste upaljeno crveno svjetlo, na semaforu glavne ceste je upaljeno zeleno, te na sporednoj

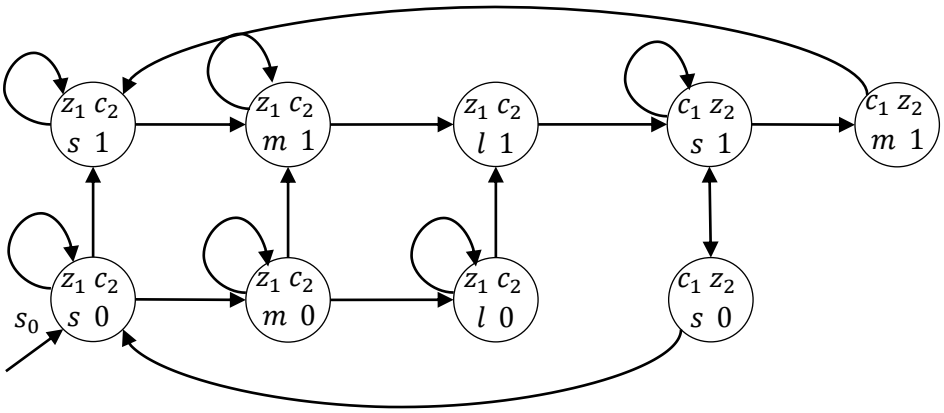
cesti ima vozila i trenutno svjetlo semafora na glavnoj cesti je upaljeno dulji vremenski period (*long*).

Svojstva koja želimo da naš sustav zadovoljava možemo sumirati sljedećom tablicom:

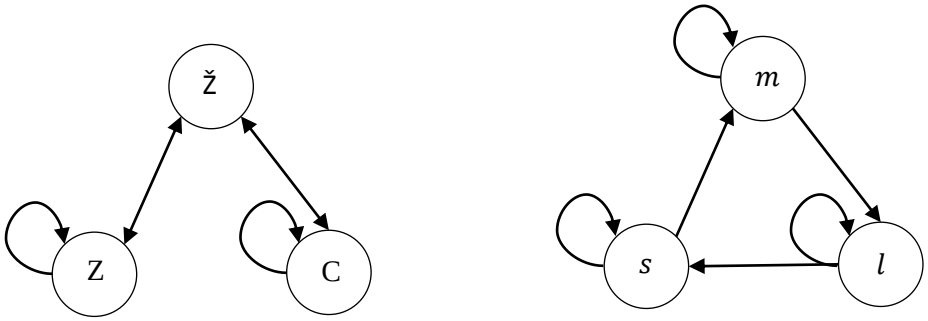
LTL formula	objašnjenje
$G\neg(z_1 \wedge z_2)$	ni u jednom trenutku ne smiju biti upaljena oba zelena svjetla
$G(z_1 \wedge l \wedge 1 \rightarrow Xz_2)$	ako je zeleno svjetlo na sporednom semaforu upaljeno duži vremenski period i pri tome na sporednom semforu čekaju auti, tada se na sporednom semaforu pali zeleno svjetlo
$G\big((c_2 \wedge Xz_2) \rightarrow z_1 \wedge l \wedge 1\big)$	zeleno svjetlo na sporednoj cesti se može upaliti samo ako je na glavnoj cesti zeleno svjetlo bilo upaljeno dulji period vremena i na sporednoj cesti ima vozila
$G\neg(c_1 \wedge l)$	crveno svjetlo nikad nije upaljeno dulji vremenski period na glavnoj cesti
$G(c_1 \wedge 0 \rightarrow Xz_1)$	ako na sporednoj cesti nema auta i na glavnom semaforu je upaljeno crveno svjetlo, onda se pali zeleno svjetlo na glavnom semaforu
$G(1 \rightarrow Fz_2)$	ovo je jedno svojstvo životnosti kojim želimo osigurati da ukoliko postoje vozila na sporednoj cesti, tada će zeleno svjetlo na semaforu sporedne ceste eventualno biti upaljeno
GFz_1	još jedno svojstvo životnosti kojim osiguravamo da će se zeleno svjetlo na glavnom semaforu uvijek eventualno upaliti

TABLICA 2.1: SVOJSTVA SUSTAVA ZA KONTROLU RADA SEMAFORA

Možemo primijetiti da sustav zadan modelom sa slike 2.4 (na sljedećoj strani) zadovoljava željena svojstva, pri čemu pretpostavljamo da nisu dozvoljeni putevi u kojima se neko stanje beskonačno mnogo puta uzastopno ponavlja. Kao što smo u prvom primjeru napomenuli, ovaj uvjet možemo osigurati uključivanjem uvjeta pravednosti u alatu za provjeru modela. Također, primijetimo da u sklopu našeg modela kontrole rada semafora možemo razlikovati četiri različita procesa koji se paralelno izvršavaju: dva semafora, sat i senzor. U realnim situacijama, model sa slike 2.4 se dobiva kombiniranjem modela pojedinih procesa. Npr. na slici 2.5 možemo vidjeti modele za semafore i sat.



SLIKA 2.4: MODEL KONTROLE RADA SEMAFORA



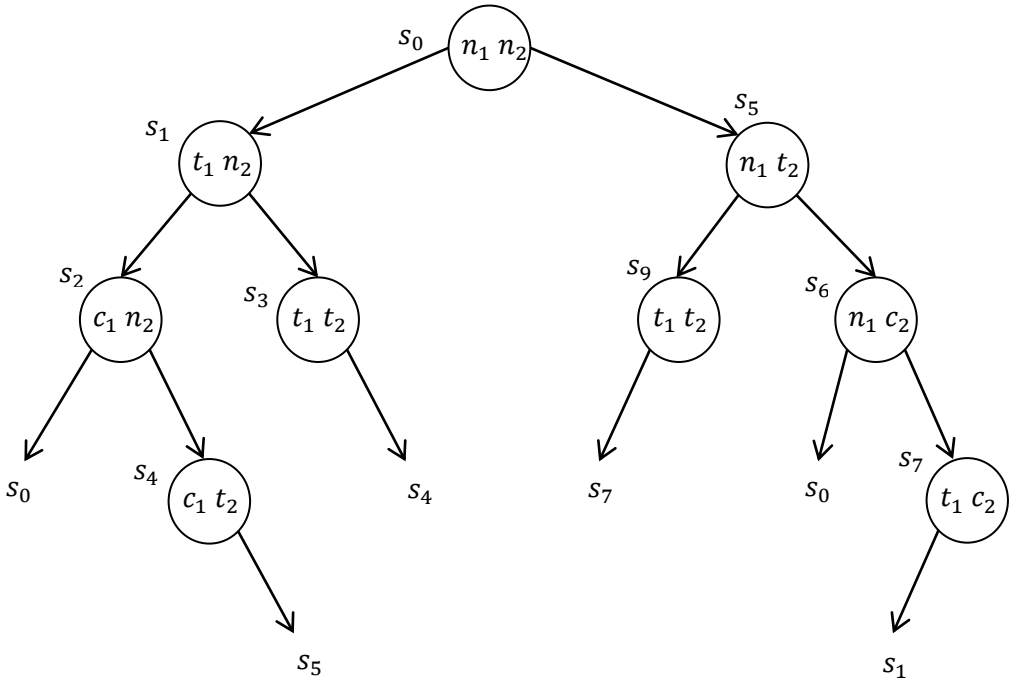
SLIKA 2.5: MODELI SEMAFORA (LIJEVO) I SATA (DESNO). MODEL SEMAFORA UKLJUČUJE I ŽUTO SVJETLO.

2.2.2 Algoritmi za provjeru modela

U pojednostavljenim primjerima iz prethodne točke mogli smo manualno provjeriti istinitost formula u početnom stanju modela. No, u realnim situacijama često se pojavljuju puno složeniji modeli za koje nije tako jednostavno provjeriti zadovoljavaju li određena svojstva. U praksi se za samu provjeru da li je određena LTL formula istinita na danom modelu koristi jedan od tzv. alata za provjeru modela (eng. *model checker*). Postoje različiti alati, a neki od popularnijih su 'New Symbolic Model Verifier' (skraćeno NuSMV) (vidi [13]) i SPIN model-checker (vidi [11]). Alati se razlikuju po deskriptivnim jezicima koje koriste za opisivanje modela i po algoritmima koje koriste za utvrđivanje istinitosti formule.

Kao što smo u točki 2.1. napominjali, vrsta algoritma za provjeru modela koji se upotrebljava ovisi i o vrsti svojstva koje promatrana formula ϕ izražava. Npr. uzmimo da formula čiju istinitost želimo provjeriti izražava jedno invarijantno svojstvo, tj. formula je oblika $G\psi$, pri čemu ψ ne sadrži temporalne operatore. Uzmimo konkretno da

je ψ negacija konjunkcije propozicionalnih varijabli c_1 i c_2. Tada se ispitivanje istinitosti formule $G\neg(c_1 \wedge c_2)$ u nekom stanju s promatranog modela $\mathcal{M}$, svodi na ispitivanje da li postoji stanje dostižno iz stanja s u modelu $\mathcal{M}$, koje je označeno varijablama c_1 i c_2. Generiranjem skupa svih dostižnih stanja dobivamo jedno stablo. Npr. na slici 2.6 je prikazano stablo koje dobijemo generiranjem skupa svih dostižnih stanja iz stanja s_0 u modelu sa slike 2.2 iz primjera 2.1 u prethodnoj točki (str. 53).



SLIKA 2.6: STABLO DOSTIŽNIH STANJA

Znači, algoritam za provjeru modela se u slučaju invarijantnih formula svodi na pretraživanje stabala. Za pretraživanje stabala se koriste DFS (*depth-first search*) i BFS (*breadth-first search*) algoritmi. DFS algoritam, koji pretražuje stablo po granama, se koristi ako se procijenjuje da bi traženo stanje moglo biti dublje u stablu. BFS algoritam, koji pretražuje stablo po nivoima, se koristi ako se procijenjuje da bi traženo stanje moglo biti blizu korijena.

Stablo sa slike 2.6 ne sadrži puno čvorova i može se brzo generirati. No, u realnim situacijama stabla dostižnih stanja mogu biti jako velika. Stoga se obično stabla ne generiraju unaprijed već se čvorovi generiraju *u hodu* (eng. on-the-fly), po potrebi kako se provjeravaju.

Za provjeravanje ostalih svojstava postoje različiti algoritmi i obično su znatno složeniji nego opisani algoritam za provjeravanje invarijantnih svojstava. No, osnovna strategija većine tih algoritama se zasniva na sljedećim koracima:

1. Za promatranu LTL formulu ϕ se konstruira Büchijev automat $A_{\neg\phi}$ koji definira svojstvo koje izražava formula $\neg\phi$. Znači, za put π u promatranom modelu $\mathcal{M}$, $A_{\neg\phi}$ prihvaća niz stanja koji odgovara putu π ako i samo ako vrijedi $\pi \vDash \neg\phi$.
2. *Kombiniranjem* modela $\mathcal{M}$ i automata $A_{\neg\phi}$ dobivamo tranzicijski sustav čiji putevi zadovoljavaju automat $A_{\neg\phi}$ a ujedno su putevi i u modelu $\mathcal{M}$.
3. Za promatrano stanje s *utvrđujemo* postoji li put iz stanja s u tranzicijskom sustavu dobivenom kombiniranjem u koraku 2. Ako takav put postoji onda na tom putu formula ϕ nije istinita i vrijedi $\mathcal{M}, s \nvDash \phi$. Inače, ako u dobivenom tranzicijskom sustavu ne postoji put iz stanja s onda vrijedi $\mathcal{M}, s \vDash \phi$.

Općenito, pokazuje se da je problem provjere modela za LTL[7] PSPACE-potpun (vidi [23]). Ispočetka se LTL nije koristio za provjeru modela, budući da postoje druge temporalne logike (točnije CTL (vidi [26])) za koje je problem provjere modela manje složenosti. No, poslije se pokazalo da postoje algoritmi za provjeru modela u LTL-u složenosti $2^{O(|\phi|)} \cdot O(|\mathcal{M}|)$ (vidi [15], [26]). U praktičnim primjenama formule koje se provjeravaju su relativno kratke i imaju neznatan utjecaj na složenost algoritama za provjeru modela, tako da je efektivna složenost algoritama iz [15], [26] zapravo linearna u veličini modela. Stoga se danas smatra da je i upotreba LTL-a pogodna za provjeru modela.

[7] Za danu LTL formulu ϕ i model $\mathcal{M}$, vrijedi li $\mathcal{M} \vDash \phi$?

Bibliografija

[1] B. Alpern, F.B. Schneider, *Defining liveness*, Information processing letters, 21 (1985), pp. 181-185.

[2] B. Alpern, F.B. Schneider, *Recognizing safety and liveness*, TR 86-727, Computer Science Department, Cornell University, 1986.

[3] M. Ben-Ari, *Mathematical Logic for Computer Science*, Springer, 2012.

[4] E.M. Clarke, E.A. Emerson, A.P. Sistla, *Automatic verication of finite state concurrent systems using temporal logics specifications* , ACM Transactions on Programming Languages and Systems, 1986.

[5] V.Diekert, P.Gastin, *First order definable languages*, Texts in Logic and Games, vol.2, Amsterdam University Press, 2008, pp. 261–306.

[6] D.M. Gabbay, A. Pnuelli, S. Shelah, J.Stavi, *On the temporal basis of fairness,* in *POPL*, ACM Press, 1980, pp. 163-173.

[7] D.M. Gabbay, *The declarative past and imperative future: Executable temporal logic for interactive systems,* in *Temporal Logic in Specification*, vol. 398. Springer, 1987, pp. 409-448.

[8] R.Gerth, D.Peled, M.Y.Vardi, P.Wolper, *Simple on-the-fly automatic verification of linear temporal logic*, Proceedings of the Fifteenth IFIPWG 6.1 International Symposiumon Protocol Specification, Testing and Verification XV, Chapman&Hall, Ltd., London, UK, 1996, pp. 3–18.

[9] T. Glavaš, $\omega-automati,$ diplomski rad, PMF-MO, Zagreb, 2013.

[10] I.M. Hodkinson, M. Reynolds, *Separation – past, present, and future,* in *We Will Show Them!* (2), College Publications, 2005, pp. 117-142.

[11] G.J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*, Addison-Wesley, Boston, MA, 2004.

[12] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory*, Languages and Computation, Addison-Wesley, 1979.

[13] M. Huth, M. Ryan, *Logic in Computer Science,* Cambridge University Press, 2004.

[14] L. Lamport, *Proving Correctness of Multiprocess Programs*, IEEE Transactions on Software Engineering, SE-3, 2 (1977), pp. 125-143.

[15] O. Lichtenstein, A. Pnueli, *Checking that finite state concurrent programs satisfy their linear specification*, ACM, POPL, 1985.

[16] N. Markey, *Temporal logic with past is exponentially more succinct, Bulletin of the EATCS*, vol. 79, 2003, pp. 122-128.

[17] R. McNaughton, S. A. Papert, *Counter-Free Automata,* (M.I.T.research monograph no. 65), The MIT Press, 1971.

[18] G. Petric Maretić, M. Torabi Dashti, D. Basin, *Anchored LTL Separation,* in *CSL-LICS*, ACM, 2014, article no. 74.

[19] G. Petric Maretić, M. Torabi Dashti, D. Basin, *LTL is closed under topological closure*, Information Processing Letters 114, pp. 408–413, 2014.

[20] M. Sipser, *Introduction to the theory of computation*, PWS Publ.Comp, 1997.

[21] A.P. Sistla, *Theoretical issues in the Design and Verification of Distributed Systems*, Ph.D. thesis 1983, Harvard University

[22] A.P. Sistla, *On characterization of safety and liveness properties in temporal logic*, in *PODC,* ACM, 1985., pp. 39-48.

[23] A.P. Sistla, E.M. Clarke, *The complexity of propositional linear temporal logics*, Journal of the ACM, 1985.

[24] R. Škorić, *Temporalna logika*, diplomski rad, PMF-MO, Zagreb, 2009.

[25] W. Thomas, *Automata on infinite objects*, in Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), J. Van Leeuwen, Ed. Elsevier and MIT Press, 1990, pp. 133–192.

[26] M.Y. Vardi, P. Wolper, *An automata theoretic approach to automatic program verication*, LICS, 1986.

[27] M. Vuković, *Matematička logika,* Element, Zagreb, 2009.

[28] P.Wolper, *Temporal logic can be more expressive*, Inf.Control 56(1/2) (1983), pp. 72–99.

.

Sažetak

U ovom diplomskom radu se upoznajemo s jednom vrstom temporalne logike koja vrijeme modelira linearno, kao niz vremenskih trenutaka izomorfan skupu $\mathbb{N}$. Diplomski rad je podijeljen u dvije cjeline: 'Jezik LTL-a' i 'Formalna verifikacija'.

U prvoj cjelini definiramo jezik LTL-a, tj. njegovu sintaksu i semantiku. Ugrubo, jezik LTL-a možemo opisati kao jezik klasične logike sudova s komponentom vremena, pri čemu se statički način interpretiranja formula zamjenjuje sa jednim dinamičkim, u kojem se istinitost formule može mijenjati kroz vrijeme.

Nakon što definiramo sintaksu i semantiku LTL-a, u sklopu prve cjeline još promatramo i jedno proširenje LTL-a (tzv. $PLTL$), u kojem postoje i temporalni operatori prošlosti. Pokazuje se da dodavanjem operatora prošlosti ništa ne dobivamo na izražajnosti jezika, u kontekstu inicijalne ekvivalencije. Znači, za svaku formulu proširenog LTL-a ϕ postoji LTL formula koja je ekvivalentna sa ϕ u početnom trenutku. Ovaj rezultat se lagano izvodi korištenjem Gabbayevog teorema separacije.

Na kraju prvog poglavlja dajemo i neke rezultate vezane za ocjene složenosti transformacije $PLTL$ formula u inicijalno ekvivalentne LTL formule, odnosno razliku u veličini između $PLTL$ formula i najmanjih njima inicijalno ekvivalentnih LTL formula (tzv. *raskorak u sažetosti*). Jedan od novijih rezultata je karakterizacija tih ocjena pomoću tzv. *usidrenih PLTL formula*. Pokazuje se da je složenost transformacije elementarna, odnosno raskorak u sažetosti je elementaran, ako i samo ako je složenost separacije usidrenog $PLTL$-a elementarna, odnosno raskorak u sažetosti između usidrenog $PLTL$-a i separiranog usidrenog $PLTL$-a je elementaran.

U drugom poglavlju se bavimo LTL-om u kontekstu formalne verifikacije. Poglavlje je podijeljeno na teorijski i praktični dio. U teorijskom dijelu razmatramo karakterizacije različitih vrsta svojstava, tj. sintaktičke karakterizacije LTL formula koje izražavaju određenu vrstu svojstava. Svojstva u kontekstu modela zapravo odgovaraju skupovima puteva. Modeli u formalnoj verifikaciji predstavljaju apstrakcije sustava koje verificiramo. Pomoću LTL formula izražavamo svojstva koja želimo da promatrani sustav zadovoljava.

Također, pokazujemo da je LTL zatvoren za topološko zatvorenje, tj. za svako svojstvo koje možemo izraziti LTL formulom znamo da se i najmanje svojstvo sigurnosti koje sadrži to svojstvo može izraziti LTL formulom. Kao posljedicu dobivamo da LTL prihvaća dekompoziciju na sigurnost i životnost (eng. *safety-liveness decomposition*), iz čega slijedi da se svaka LTL formula može prikazati kao konjunkcija formule sigurnosti i formule životnosti.

U praktičnom dijelu drugog poglavlja kroz primjere pokazujemo kako se LTL konkretno primjenjuje u sklopu jedne metode za verifikaciju, tzv. provjere modela (eng. *model checking*). LTL provjera modela se svodi na ispitivanje da li je za dani model $\mathcal{M}$ i

početno stanje s_0, određena LTL formula ϕ istinita na svim putevima iz stanja s_0 u modelu $\mathcal{M}$.

Glavna literatura za diplomski rad je Ben-Arijeva knjiga ([3]), ali značajniji utjecaj na sadržaj diplomskog rada su imali i članci Grugura Petric-Maretića ([18],[19]), članak A.P. Sistle [22] te knjiga [13] autora M.Huth i M.Ryan.

Htio bih se zahvaliti mojim roditeljima na podršci. Također, htio bih se zahvaliti Grguru Petric-Maretiću s doktorskog studija fakulteta ETH u Zürichu za pomoć pri nabavljanju literature te razjašnjavanju nekih nedoumica vezanih za usidreni $PLTL$. Posebno, htio bih se zahvaliti mojem mentoru, prof.dr.sc. Mladenu Vukoviću koji je bio vrlo pristupačan i susretljiv, te od neizmjerne pomoći prilikom izrade ovog diplomskog rada.

Summary

In this thesis we get acquainted with a type of temporal logic that has a linear model of time. Time is modeled as a set of sequences of moments isomorphic with $\mathbb{N}$. Roughly, we could describe LTL as classical propositional logic with the ability to express time. In LTL time is expressed through temporal operators X, F, W, U, G and R. Another peculiarity of LTL is that all of the temporal operators only refer to the future. In LTL we cannot refer to the past. A more pronounced difference between LTL and classical propostional logic is in the way formulas are interpreted. In LTL formulas are interpreted with the use of *transitional systems,* which are also called *models*. More precisely, LTL formulas are interpreted on sequences of *states* of a transitional system, that are labeled with propostional variables. Such sequences are called *paths*. If a state is labled with a certain variable, then it means that the propositional variable is true in that state. In other words, the static notion of truth in classical propostitional logic is replaced with a dynamic one, in which the formulas may change their truth values as the system evolves from state to state.

As we already mentioned, in LTL there are no operators with which we could refer to the past. By studying an expansion of LTL (so called $PLTL$), that has operators that refer to the past, we see that by adding these operators we don't get anything as far as expressiveness is concerned, within the context of initial equivalence. So, for every $PLTL$ formula ϕ there is an LTL formula that's equivalent to ϕ in the initial moment. This result can be easily deduced by using Gabbay's separation theorem. Separation is a fundamental concept for temporal logics, which was first introduced by Gabbay. Roughly, a temporal logic has the separation property if for every formula of that logic is an equivalent boolean combination of a future, past and present formula. By Gabbay's separation theorem we know that $PLTL$ has the separation property.

There are various results that describe the complexity of transforming $PLTL$ formulas into initially equivalent LTL formulas and the difference in size between $PLTL$ formulas and the shortest initally equivalent LTL formulas (this difference is also called the *succinctness* gap). One of the newer results is a characterization that we get by observing a strict subset of $PLTL$ formulas that are called *anchored* $PLTL$ formulas. It is shown that the complexity of the transformation is elementary if and only if the complexity of anchored $PLTL$ separation is elementary. Also, the succinctness gap between LTL and $PLTL$ is elementary if and only if the succinctness gap between anchored $PLTL$ and seperated anchored $PLTL$ is elementary.

Linear temporal logic has shown to be a suitable logic for formal verification of systems. Formal verification is very important in the industry for determining the correctness of, so called, *critical* hardware\software systems. There are various formal verification methods. One of the more popular for use is *model checking*. For a model $\mathcal{M}$,

initial state s_0 and an LTL formula ϕ, LTL model checking comes down to determining whether the formula ϕ is true on all paths beggining at s_0 in model $\mathcal{M}$.

The models in formal verification represent abstractions of systems that need to be verified. With LTL formulas we express properties of the system that we want to check. There are different algorithms for model checking. The type of the property that we want to check has a significant influence on the way we shape the process of verification and on the selection of the algorithm for model checking. Within the context of models, properties are sets of paths. We distinguish different types of properties, some of which are safety properties, liveness properties, fairness... Because the type of the property influences the way systems are verified, it is important to determine the type of the property we want to check. For this purpose, characterization of properties, that is syntactic characterizations of LTL formulas that express a certain type of property, can be very helpful. In section 2.1. we give characterizations of different types of properties.

Also, in this section we show that LTL is closed for topological closure, that is for every property that we can express with an LTL formula the smallest safety property that contains it can also be expressed with an LTL formula. Corollarily we get that LTL admits the saftey-liveness decomposition, which means that every LTL formula is equivalent to a conjunction of a safety formula and a liveness formula.

The main literature for this thesis was Ben-Ari's book ([3]), but a significant contribution to the content of the thesis was also drawn from Grgur Petric-Maretic's articles ([18],[19]), A.P. Sistla's article [22] and the book [13] written by authors M.Huth and M.Ryan.

Životopis

Rođen sam 15.7.1984.g. u Doboju, BiH. U Doboju upisujem i završavam prvi razred osnovne škole. Početkom rata 1992.g. selim se u Sisak. Drugi i treći razred pohađam u OŠ 22.lipnja u Sisku. Nakon završetka trećeg razreda 1994.g., selim se u Karlovac. U Karlovcu se upisujem u OŠ Grabrik, koju završavam 1998.g. Iste godine upisujem Opću Gimnaziju Karlovac, koju uspješno završavam 2003.g. Nakon završetka gimnazije, upisujem se na Matematički odsjek PMF-a u Zagrebu. Preddiplomski sveučilišni studij Matematike završavam 2010.g. i time stječem akademski naziv sveučilišnog prvostupnika Matematike. Iste godine, nakon završetka preddiplomskog studija, upisujem diplomski sveučilišni studij Računarstvo i matematika na Matematičkom odsjeku PMF-a.

Otac Anto je umirovljeni profesor povijesti, a majka Iva je umirovljena profesorica matematike. Slijedeći primjer roditelja, od početka školovanja najviše interesa pokazujem za povijest i matematiku. Također, od ranih dana školovanja pokazujem sklonost prema stranim jezicima te pohađam tečajeve za engleski jezik. No, već pri kraju osnovne škole postaje jasno da me matematika ipak najviše interesira. Tokom osnovnoškolskog i srednjoškolskog obrazovanja također pokazujem interes za sportske aktivnosti. Treniram razne sportove: nogomet, veslanje, stolni tenis, a najduže tenis (sedam godina). U srednjoj školi s odličnim uspjehom polažem predmete Matematiku i Informatiku sve četiri godine te ostvarujem pravo izravnog upisa na PMF-MO.

Početkom studija selim se u Zagreb i počinjem samostalno živjeti. Prvih nekoliko godina relativno uspješno obavljam studentske obveze na fakultetu. No, pri kraju preddiplomskog studija počinjem raditi honorarne poslove preko Studentskog servisa. Vođen zaradom koju ostvarujem obavljanjem raznoraznih studentskih poslova zapostavljam studentske obveze. Nekako privodim kraju preddiplomski studij, ali postaje upitan nastavak studiranja. Ipak, na kraju prevladava moja želja za daljnjim obrazovanjem. Prestajem raditi studentske poslove te se posvećujem završavanju diplomskog studija, nakon čega redovno polažem ispite te uspješno ispunjavam studentske obveze.